

AD-A180 344

SYSTEMS ARCHITECTS INC RANDOLPH MASS

F/G 5/2

COMPUTER SYSTEMS ACQUISITION METRICS HANDBOOK, VOLUME III, DATA--ETC (U)
MAY 82

F19628-80-C-0207

UNCLASSIFIED

ESD-TR-82-143(3)

NL

122

100

E

3

1

AD A120377

ESD-TR-82-143(III)

DATA ELEMENT DICTIONARY FOR COMPUTER
SYSTEMS ACQUISITION METRICS HANDBOOK.
VOLUME III.

Systems Architects, Inc.
50 Thomas Patten Drive
Randolph, MA 02368

May 1982

Approved for public release;
Distribution Unlimited.



DTIC
ELECTE
OCT 18 1982
H

DTIC FILE COPY

Prepared for

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
DEPUTY FOR TECHNICAL OPERATIONS AND
PRODUCT ASSURANCE
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731

82 10 18 071

LEGAL NOTICE

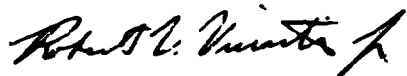
When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.


OTHER NOTICES

Do not return this copy. Retain or destroy.

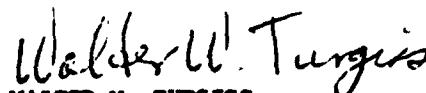
REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.


ROBERT V. VIERAITIS, Jr., 1Lt, USAF
Project Officer


JAMES W. NEELY, Jr., Lt Col, USAF
Chief, Computer Engineering
Applications Division

FOR THE COMMANDER


WALTER W. TURGISS
Acting Director, Engineering and Test
Deputy for Technical Operations
and Product Assurance

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|-------------------------------------|--|
| 1. REPORT NUMBER ESD-TR-82-143 (III) | 2. GOVT ACCESSION NO. AD-A120377 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Data Element Dictionary for Computer Systems Acquisition Metrics Handbook. Volume III. | | 5. TYPE OF REPORT & PERIOD COVERED |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Systems Architects, Inc. | | 8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0207 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Systems Architects, Inc 50 Thomas Patten Drive Randolph, MA 02368 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Electronic Systems Division (TOEE) Hanscom AFB Massachusetts 01731 | | 12. REPORT DATE May 1982 |
| | | 13. NUMBER OF PAGES 148 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |
| 16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. | | |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) | | |
| 18. SUPPLEMENTARY NOTES | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer systems Metrics Quality assurance Software | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Data Element Dictionary is a reference guide for all the data elements. It lists the name, index number, life cycle phase description, an example with explanations and worksheet reference for each data element. | | |

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

NAME: Unambiguous References

INDEX NUMBER: 2

DATA ELEMENT: Are requirements itemized so that the various functions to be performed, their inputs and outputs, are clearly delineated?

Y / N

LIFE CYCLE PHASE(S):

- (1) Requirement Analysis
- (2) Preliminary Design
- (3) Detail Design

DESCRIPTION: Unique references to data or functions avoid ambiguities such as a function being called one name by one module and by another name by another module. Unique references avoid this type of ambiguity in all three phases.

EXAMPLE: Function Name: Radar Prediction Processor

Inputs:

Outputs:

Function:

Function Name: Interactive Display Module

Inputs:

Outputs:

Function:

Function Name: Prediction Recording Module

Inputs:

Outputs:

Function:

EXPLANATION: Each system function, or subsystem module or subroutine should have its inputs, outputs, and function clearly outlined and described beginning in the Requirements Analysis phase, and strictly adhered to in the design and implementation phases.



| | | | | | | | | |
|---------------|-----------|----------|-------------|---------------|----|--------------|--------------------|-----------------------------|
| Accession For | DTIC 0141 | DTIC 225 | Unannounced | Justification | By | Distribution | Availability Codes | Dist. Avail. and/or Special |
| | | | | | | | | A |

This is a binary measure which may be answered with a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoRAM.1
CoPDM.1
CoDDM.3

Co-16
Co-20
Co-34

NAME: External Data Reference

INDEX NUMBER: 3

DATA ELEMENT: Number of data references which are defined.

Number of major data references.

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design
(3) Detail Design

DESCRIPTION: Each data element is to have a specific origin. At the requirements level only, major global data elements and a few specific local data elements may be available to be checked. The set of data elements available for completeness checking at the design level increases substantially and is to be completed at implementation.

EXAMPLE: All data items referenced in XALTCLTB structured flowcharts are defined in the following list

| <u>DATA ITEM DESCRIPTION</u> | <u>DEFINITION</u> |
|------------------------------|-----------------------|
| XALFMTLP | See Appendix III |
| XZDNTRAC | See CNCE-C5-478T-0023 |
| XZABKEY | See Appendix III |
| XNAPIPP | See Appendix III |
| XZATFLAG | See Appendix III |
| XNAVDU | See Appendix III |
| XZACFLAG | See Appendix III |
| XZAMAXPG | See Appendix III |
| XZAPFLAG | See Appendix III |
| XALD01DCB | See Appendix III |
| XALD01BUF | See Appendix III |
| RECKEY | See Appendix III |
| STATUS | See Appendix III |
| XALROW | See Appendix III |
| XALCOL | See Appendix III |
| XZAXY | See Appendix III |
| XALAKEY1 | See Appendix III |
| XALVFLD1 | See Appendix III |
| XZDNWORK | See Appendix III |

EXPLANATION:

This score is obtained by dividing the number of defined data references by the number of major data references. If any data names other than those above where referenced in the code, they will be undefined.

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoRAM.1

Co-16

CoPDM.1

Co-20

CoDDM.3

Co-34

NAME: Major Functions Used

INDEX NUMBER: 4

DATA ELEMENT: Number of defined functions used.

Number of major functions identified

LIFE CYCLE PHASE(S):

- (1) Requirement Analysis
- (2) Preliminary Design
- (3) Detail Design

DESCRIPTION: A function which is defined but not used during a phase is either nonfunctional or a reference to it has been omitted.

EXAMPLE: A function that is defined is a function that exists somewhere in the documentation and which contains a description of its usage, process, limitations and the inputs to and outputs from that function. If that function is never called by another module it is unused. Therefore, if module GET is defined as a function of a system, but no other module calls it, GET is an unused module.

EXPLANATION: This score is obtained by dividing the number of major functions used by the total number of major functions identified.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | CoRAM.1 | Co-16 |
| | CoPDM.1 | Co-20 |
| | CoDDM.3 | Co-34 |

NAME: Major Functions Defined

INDEX NUMBER: 5

DATA ELEMENT: Number of identified functions defined.
Number of functions identified.

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design
(3) Detail Design

DESCRIPTION: A system is not complete at any phase if dummy functions are present or if functions have been referenced but not defined.

EXAMPLE: A system may contain a reference or call to a function/module that has never been defined (i.e., there is no description of its inputs, outputs, process, limitations, etc, and no design for the function has been performed). This function/module has been identified, but has not been defined. For example, a system has 6 major functions, all of which have been thoroughly defined and which call each other, therefore they have all been identified. One of these functions also references another function, as yet undefined, called PUSH. At this time only dummy parameters are passed in and out of this module.

EXPLANATION: This score is obtained by dividing the number of major functions defined by the major identified.

The # of functions identified = 7
of functions defined = 6
Therefore, the score would be $6/7 = .86$

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | CoRAM.1 | Co-16 |
| | CoPDM.1 | Co-20 |
| | CoDDM.3 | Co-34 |

NAME: Decision Points Defined

INDEX NUMBER: 6

DATA ELEMENT: Is the flow of processing and all decision points in that flow defined? Y / N

LIFE CYCLE PHASE(S):

- (1) Requirement Analysis
- (2) Preliminary Design
- (3) Detail Design

DESCRIPTION: Each decision point is to have all of its conditions and alternative processing paths defined at each phase of the software development. The level of detail to which the conditions and alternative processing are described may vary, but the important element is that all alternatives are described.

EXAMPLE: Flowcharts following the conditions and processing paths of decision points are an example of decision point description

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | CoRAM.1 | Co-17 |
| | CoPDM.1 | Co-21 |
| | CoDDM.3 | Co-35 |

NAME: Agreement of Calling Sequence Parameters

INDEX NUMBER: 7

DATA ELEMENT: Number of defined and referenced calling sequence parameters that agree between functions.

Number of calling sequence parameters.

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design
(3) Detail Design

DESCRIPTION: For each interaction between modules, the full complement of defined parameters for the interface is to be used. A particular call to a module should not pass, for example, only five of the six defined parameters for that module.

EXAMPLE: Main Program

DIMENSION A(3,3),B(3,3),C(3,3)
DIMENSION T(2,2),Q(2,2),R(2,2)
M=3
N=3
CALL ADD(A,B,C,M,N)

Subprogram

```
SUBROUTINE ADD(X,Y,Z,M,N)
  DIMENSION X(M,N),Y(M,N),Z(M,N)
  DO 10 I=1,M
    DO 10 J=1,N
10  Z(I,J)=X(I,J)+Y(I,J)
  RETURN
END
```

EXPLANATION: This score is obtained by dividing the number of parameters passed by the total number of parameters.

The call to the subroutine "ADD" contains the same number of parameters as defined in the subroutine statement. The types of the parameters also agree (i.e., both integer or both real, etc).

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | CoRAM.1 | Co-17 |
| | CoPDM.1 | Co-21 |
| | CoDDM.3 | Co-35 |

NAME: Problem Reports Resolved

INDEX NUMBER: 8

DATA ELEMENT: Number of those problem reports that have been closed (resolved).

Number of problem reports related to the requirements that have been reported.

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design
(3) Detail Design

DESCRIPTION: At each phase in the development, problem reports are generated. Each problem report is to be closed or the resolution indicated so that a complete product can be ensured.

EXAMPLE: Problem reports may include: (1) Computational, (2) Logic, (3) Input/Output, (4) Data Handling, (5) Operating System Support, (6) Configuration, (7) Routine/Routine Interface, (8) Routine/System Interface, (9) Tape Processing, (10) User Interface, (11) Data Base Interface, (12) User Requested Changes, (13) Preset Data, (14) Global Variable Definition, (15) Recurrent Error, (16) Documentation, (17) Requirement Compliance, (18) Operator, (19) Questions, and (20) Hardware. A report form is submitted closing 18 of the reports.

EXPLANATION: The number of problem reports related to the requirements that have been recorded is twenty (20). The number of problem reports that have been resolved is eighteen (18). This score is obtained by dividing the number of problem reports resolved by the number of problem reports.

Score = 18/20
= 0.90.

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoRAM.1
CoPDM.1
CoDDM.3

Co-17
Co-21
Co-35

NAME: Error Analysis

INDEX NUMBER: 9

DATA ELEMENT: Has an error analysis been performed
and budgeted to functions?

Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: An error analysis must be part of the requirements analysis performed to develop the requirement specification. This analysis allocates overall accuracy requirements to the individual functions to be performed by the system. This budgeting of accuracy requirements provides definitive objectives to the module designers and implementers.

EXAMPLE: Requirement Analysis

| System Functions | Allowed Error approximation |
|------------------|-----------------------------|
| Subsystem1 | $\pm 1\%$ |
| Subsystem2 | $\pm 5\%$ |
| Subsystem3 | $\pm 3\%$ |
| Subsystem4 | $\pm 8\%$ |

EXPLANATION: There should be some sort of delineations of the accuracy requirements for each subsystem or function in the system.

This is a binary measure answered by a "Yes or "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReRAM. 5

Re-20

NAME: Accuracy Requirement

INDEX NUMBER: 10

DATA ELEMENT: Are there definitive statements of the accuracy requirements for inputs, processing, and constants? Y/N

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: A definitive statement of accuracy requirements must be part of the requirements analysis performed to develop the requirement specification. The accuracy requirement should concern the inputs, outputs, processing, and constants for each individual function.

EXAMPLE: How accurate must the inputs be to each subroutine or function? Can the inputs be integers, or must they be real numbers? How many decimal places will be required on the output? What happens if an overflow condition occurs? Will stars be outputted, or will inaccurate results be included on output? These and related considerations should be included in the accuracy requirements.

EXPLANATION: In the Requirement Analysis Phase, a statement should be included for each major function requiring each of the above conditions to be met.

This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReRAM.5 | Re-20 |

NAME: Error Tolerance of Input Data

INDEX NUMBER: 11

DATA ELEMENT: Are there definitive statements of
the error tolerance for input data?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: The requirement specification must identify the
desired error tolerance capabilities.

EXAMPLE: The requirements specification should specify
what input constitutes a fatal error, what input
constitutes a warning and what types of errors
are passable without halting processing

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReRAM.1

Re-16

NAME: Recovery from Computational Failures

INDEX NUMBER: 12

DATA ELEMENT: Are there definitive statements of
the requirements for recovery from
computational failures?

Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: The requirement for this type of error tolerance
capability must be stated during the Require-
ments Analysis Phase.

EXAMPLE: On recovery of computational errors, checks must
be made to resume normal operations, insure that
data base elements have not been damaged and
computations have not been affected and the
processing remains accurate

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReRAM.2

Re-17

NAME: Statements of Recovery From Hardware Faults

INDEX NUMBER: 13

DATA ELEMENT: Is there a definitive statement of
the requirement for recovery from
hardware faults?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: The handling of hardware faults such as arithmetic faults, power failure, and clock interrupts must be stated during the Requirements Analysis Phase.

EXAMPLE: Recovery from hardware faults is expressed in the amount of time required for the system to be restored after a hardware failure. It should also include the correction of any data affected, damaged, or destroyed as a result of the hardware failure

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReRAM.3 | Re-18 |

NAME: Statements of Recovery From Device Errors

INDEX NUMBER: 14

DATA ELEMENT: Is there a definitive statement of
the requirement for recovery from
device errors?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: The handling of device errors such as expected
end-of-files or end-of-tape conditions or read/
write failures must be stated during the Re-
quirements Analysis Phase.

EXAMPLE: If data is being written from disk to tape and
an end-of-tape is reached that is unexpected
what procedure is followed? Does the processing
hang up and wait for a new tape to be mounted?
Is a message sent to the operator? Or, perhaps
an operating system providing a check-point
restart capability.

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReRAM.4 | Re-19 |

NAME: Input/Output Access Control

INDEX NUMBER: 15

DATA ELEMENT: Is there a definitive statement of the requirement for user input/output access controls in the Requirements Analysis Phase? Y / N

Are user input/output access controls provided in the Design and Implementation Phases? Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: The requirements for user access control must be identified during the Requirements Analysis Phase. Provisions for identification and password checking must be designed and implemented to comply with the requirements. This binary measure is applied in all three phases of development, and identifies if attention has been placed on this area.

EXAMPLE: A system may have different levels of access. Data entry operators would have access to specific files with a password. A maintenance programmer would have access to most files while the system administrator would have access to all files. They would probably have different passwords also.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ItRAM.1 | It-15 |
| | ItPDM.1 | It-21 |

NAME: Data Base Access Control

INDEX NUMBER: 16

DATA ELEMENT: Is there a definitive statement of the requirement for data base access controls in the Requirements Analysis Phase? Y / N /

Are data base access controls provided in the Design and Implementation Phases? Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: This binary measure identifies whether requirements for data base controls have been specified and designed, and if the capabilities have been implemented.

EXAMPLE: A definitive statement of requirements for authorization tables and privacy locks is made in the requirements specification. These authorization tables and privacy locks are implemented in the Design and Implementation Phases.

EXPLANATION: This is a binary measure answered by a "Yes or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ItRAM.1 | It-15 |
| | ItPDM.1 | It-21 |

NAME: Memory Protection

INDEX NUMBER: 17

DATA ELEMENT: Is there a definitive statement of the requirement for memory protection across tasks in the Requirements Analysis Phase? Y / N

Is memory protection across tasks provided in the Design and Implementation Phases? Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: This binary measure identifies the progression from a requirement statement to implementation of memory protection across tasks.

EXAMPLE: This type of protection is often provided to some degree by the operating system. It prevents tasks from accessing system registers.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ItRAM.1 | It-15 |
| | ItPDM.1 | It-21 |

NAME: Recording and Reporting Access

INDEX NUMBER: 18

DATA ELEMENT: Are there definitive statements of the requirements for recording and reporting access in the Requirements Analysis Phase? Y / N /

Are provisions for recording and reporting access provided in the Design and Implementation Phases? Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: A statement of the requirement for this capability must exist in the requirements specification. It is to be considered in the design specification and is coded during implementation. This binary measure is applied in all three phases and identifies whether these steps are being taken.

EXAMPLE: The recording of terminal linkages, data file accesses, and jobs run by user identification and time is stated in the Requirements Phase, and is provided in the Design and Implementation Phases.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ItRAM.2 | It-16 |
| | ItPDM.2 | It-22 |

NAME: Indication of Access Violation

INDEX NUMBER: 19

DATA ELEMENT: Is there a definitive statement of the requirement for immediate indication of access violation in the Requirements Analysis Phase? Y / N /

Are provisions for indication of access violation provided in the Design and Implementation Phases? Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: Access audit capabilities required might include not only recording accesses, but immediate identification of unauthorized access, whether intentional or not. This measure traces the provision for access audit capability in the Requirements Phase and the Design and Implementation Phases.

EXAMPLE: An access violation occurs when an unauthorized entry to a protected area has been accomplished. This violation may be successful, in which case a report of the violation is probably not possible, or an attempted access violation which the operating system or other software prevents and reports.

For example, an unauthorized person trying to gain access to a data base may try guessing a password. After 3 unsuccessful attempts, a message may be sent to the operator informing him of those unsuccessful attempts and the location where they are taking place.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ItRAM.2 | It-16 |
| | ItPDM.2 | It-22 |

NAME: Operating Characteristics

INDEX NUMBER: 20

DATA ELEMENT: Are all steps in the operation described?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: This binary measure is applied in all three phases of development. It identifies whether the operating characteristics have been described in the requirement specification and if this description has been transferred into an implementable description of the operation (usually in an operator's manual). This description of the operation should cover normal sequential steps and all alternative steps.

EXAMPLE: The requirement specification should have information describing operation of the system. This information should be easily transferable to an operators manual to completely cover all steps of the operation. If the information is not complete or the transfer from the requirements specification to the operators manual is not a simple one, then the answer to this metric is "No".

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.1 | Us-15 |
| | UsPDM.1 | Us-24 |

NAME: Error Conditions and Responses

INDEX NUMBER: 21

DATA ELEMENT: Are all error conditions to be reported to the operator/user identified, and are the responses described?

/ Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: The requirement for this capability must appear in the requirement specification. It must be considered during design and coded during implementation. Error conditions must be clearly identified by the system. Legal responses for all conditions are to be either documented and/or prompted by the system. This is a binary measure to trace the evolution and implementation of these capabilities.

EXAMPLE: The requirements specification and design documents should make provisions for error conditions and responses. The system could prompt the user in the following manner: "Illegal value entered, Integer Only."

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.1 | Us-15 |
| | UsPDM.1 | Us-24 |

NAME: Operator's Capabilities

INDEX NUMBER: 22

DATA ELEMENT: Is there a definitive statement of the requirement for the capability to interrupt operation, obtain status, modify, and continue processing in the Requirements Analysis Phase? Y / N

Are provisions for the operator to interrupt operation, obtain status, modify, and continue processing provided in the Design Phase? Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: The capabilities provided to the operator must be considered during the Requirement Phase and must then be designated and implemented. This is a binary measure to trace the evolution and implementation of these capabilities.

EXAMPLE: Operator capabilities to be specified might include halt/resume and check pointing.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.1 | Us-15 |
| | UsPDM.1 | Us-24 |

NAME: Flexibility of Input

INDEX NUMBER: 23

DATA ELEMENT: Is there a definitive statement of the requirement for optional input media in the Requirements Analysis Phase? Y / N

Are provisions for input to be specified from different media provided in the Design Phase? Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: The flexibility of input must be decided during the Requirements Analysis Phase and followed through during the Design and Implementation Phases. This is a binary measure of the existence of the consideration of this capability during all three phases.

EXAMPLE: If a module reads input from disk, will it also accept input from cards, tape, etc.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.3 | Us-17 |
| | UsPDM.2 | Us-26 |

NAME: Flexibility of Output

INDEX NUMBER: 24

DATA ELEMENT: Is there a definitive statement of the requirement for optional output media in the Requirements Analysis Phase?

Y / N

Are provisions for directing output to different media provided in the Design and Implementation Phases?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: This is a binary measure which identifies if there is consideration of the capability to redirect output to different media. This is a binary measure of the existence of the consideration of this capability during all three phases.

EXAMPLE: A module may output data to disk but can it also output data to tape, hard copy, etc.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.2 | Us-16 |
| | UsPDM.3 | Us-28 |

NAME: Output Control

INDEX NUMBER: 25

DATA ELEMENT: Is there a definitive statement of
the requirement for selective output
control in the Requirements Analysis
Phase?

Y / N /

Are provisions for selective output
controls provided in the Design and
Implementation Phases?

Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Preliminary Design

DESCRIPTION: The existence of a requirement for, design for,
and implementation of selective output controls
is indicated by this binary measure.

EXAMPLE: Selective controls include choosing specific
outputs, output formats, and amount of output

EXPLANATION: This is a binary measure answered by a "Yes" or a
"No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.2 | Us-16 |
| | UsPDM.3 | Us-28 |

NAME: Performance Requirements

INDEX NUMBER: 26

DATA ELEMENT: Have performance requirements (storage and run time) been identified in the Requirements Analysis Phase for the functions to be performed? Y / N /

Are specific performance requirements allocated to this module in the Design Phase? Y / N /

LIFE CYCLE PHASE(S): (1) Requirement Analysis
(2) Detail Design

DESCRIPTION: Performance requirements for the system must be broken down and allocated approximately to the modules during the Design Phase. This metric simply identifies if the performance requirements have or have not been allocated to the design.

EXAMPLE: Detail Design Phase
Function: TABLELOOKUP
Storage requirements: 2K bytes
Run time: .03 CPU Sec.
Response time: .001 CPU Sec.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | EfRAM.1 | Ef-16 |
| | EfDDM.2 | Ef-29 |

NAME: Communication with Other Systems

INDEX NUMBER: 27

DATA ELEMENT: Is there a definitive statement of
the requirement for communication
with other systems?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: During the Requirements Analysis Phase, the
communication requirement with other systems
must be considered. This metric is a binary
measure of the existence of this consideration.

EXAMPLE: RAPS System:

Will communicate with DUNS system via PIPS
protocol

Also communicates with LUNK system via PIP3
protocol

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

IpRAM.1

Ip-16

NAME: Standard Data Representation for Communication

INDEX NUMBER: 28

DATA ELEMENT: Is there a definitive statement of
the requirement for standard data
representation for communication with
other systems?

Y / N

LIFE CYCLE PHASE(S): (1) Requirement Analysis

DESCRIPTION: This metric is a binary measure of the existence
of the consideration for standard data represen-
tation between systems which are to be inter-
faced. This must be addressed and measured
during the Requirements Analysis Phase.

EXAMPLE: (1) Identify files that the system requires
(input, intermediate, output.)
(2) Identify data structures that will be used
to process the data (arrays, lists, trees,
records, etc.)

EXPLANATION: This is a binary measure answered by a "Yes or
"No."

WORKSHEET REFERENCE:

FORM CODE

PAGE

IpRAM.2

Ip-17

NAME: Cross Reference (Traceability)

INDEX NUMBER: 29

DATA ELEMENT: Is there a matrix relating itemized requirements to modules which implement these requirements? Y / N /

LIFE CYCLE PHASE(S): (1) Preliminary Design
(2) Implementation

DESCRIPTION: The identification of the itemized requirements which are satisfied in the design of a module are documented during Design Phase.

EXAMPLE: A traceability matrix is an example of how this can be done. During the Implementation Phase, the itemized requirements which are satisfied by the module implementation are to be identified. Some form of automated notation, prologue comments, or embedded comments is used to provide this cross reference. The metric is the identification of a tracing from requirements to design to code.

Data Entry/Update Query System

| Module | Storage | Response Time |
|--------|---------|---------------|
| Entry | 3K | 5 sec |
| Update | 5K | 2 sec |
| Query | 7K | 5 sec |

EXPLANATION: This is a binary measure which demonstrates the presence or absence of a matrix relating itemized requirements to modules which implement these requirements.

| | | |
|-----------------------------|------------------|-------------|
| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
| | CoPDM.2 | Co-22 |
| | CoIMM.1 | Co-42 |

NAME: Central Control

INDEX NUMBER: 31

DATA ELEMENT: Is concurrent processing centrally controlled? Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: Functions which may be used concurrently are to be centrally controlled to provide concurrency checking and read/write locks. The central control must be considered in the Preliminary Design Phase and then implemented.

EXAMPLE: If a system consisting of a VAX11, PDP11/60, and a DEC 2060 was interfaced and allowed concurrent processing between the VAX11 and the PDP11/60, the DEC 2060 software should act as a controller to provide checking and read/write locks, etc. The situation could occur between software on the same hardware (such as a Virtual IBM OS), in that case, one system should set aside as the controller for the others.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | RePDM.1 | Re-25 |

NAME: Error Fixable and Processing Continued

INDEX NUMBER: 32

DATA ELEMENT: Number of errors that are automatically bypassed while processing continues.

Number of error conditions reported by the system.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: When an error is detected, the capability to correct the error on-line and then to continue processing should be available. This can be measured during the Design and Implementation Phases.

EXAMPLE: An example of this would be an operator message that the wrong tape is mounted and processing will continue when the correct tape is mounted.

EXPLANATION: This metric is scored by dividing the number of fixed or bypassed errors by the total number of error conditions.

WORKSHEET REFERENCE:

FORM CODE

PAGE

RePDM.1

Re-25

NAME: Provisions of Recovery From Hardware Faults

INDEX NUMBER: 33

DATA ELEMENT: Are provisions for recovery from hardware faults provided? / Y / N /

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The design specifications and source code which allow for recovery from hardware faults must exist in the Design and Implementation Phases.

EXAMPLE: If the hardware system is not equipped with a bootstrap loader, there should be a specific procedure to follow a system crash or power failure in order to get the system up again.

EXPLANATION: This is a binary measure which is answered by a "Yes" or "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | RePDM.2 | Re-26 |

NAME: Provisions of Recovery From Device Errors

INDEX NUMBER: 34

DATA ELEMENT: Are provisions for recovery from device errors provided? / Y / N /

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The design specifications and source code which allow for recovery from device errors must exist in the Design and Implementation Phases.

EXAMPLE:

- (1) Hardware may suffer from transient errors such as loose cables, malfunctions of computer components undetailed IO errors or channel errors, timing errors that cause interrupts. Or the errors may be less temporary, such as disk malfunctions, tape reader malfunctions, etc.
- (2) Instructions for recovery from each of these types of hardware errors (and others not identified here) should have been included in the Requirements Analysis Phase and then design should begin in the Preliminary Design Phase.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | RePDM.3 | Re-27 |

NAME: Hierarchical Structure

INDEX NUMBER: 35

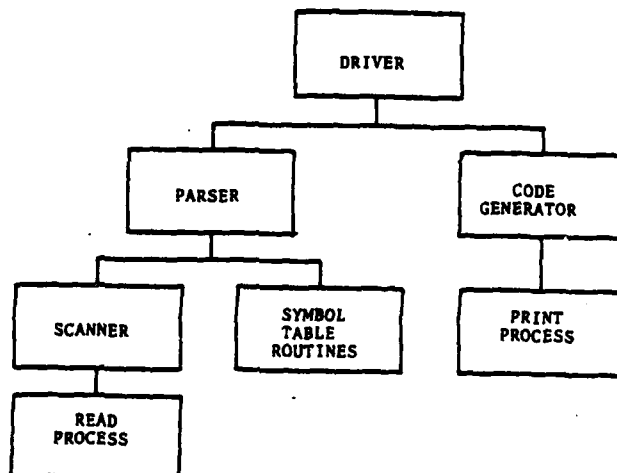
DATA ELEMENT: Is a hierarchical chart provided which identifies all modules in the system?

Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: A hierarchical chart of system modules is usually available or is easy to construct from the design documentation. It should reflect the accepted notion of top-down design, in which each level of the tree represents lower levels of detail description of the processing.

EXAMPLE:



EXPLANATION: In a real-life system, this design chart would be much more detailed - each level of this diagram would have levels of detail beneath it.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | RePDM.4 | Re-28 |
| | MaPDM.2 | Ma-17 |
| | TePDM.1 | Te-16 |

NAME: Module Independence

INDEX NUMBER: 36

DATA ELEMENT: Is the module independent of the source of the input or the destination of the output?

Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The processing done within a module is not be dependent on the source of input or the destination of the output. This rule can be applied to the module description during the Design Phase and to the coded module during the Implementation Phase.

EXAMPLE: A module should be independent of how it was invoked. Any decisions made within the module should be made independent of which module was the calling module, or what type of device was the calling device. Decisions should also be independent of what module will receive its output, or of any other process occurring outside of its bounds.

EXPLANATION: This is a binary measure which is answered by a "Yes" or "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | RePDM.4 | Re-28 |
| | MaPDM.2 | Ma-17 |
| | TePDM.1 | Te-16 |

NAME: Storage Allocation Requirements

INDEX NUMBER: 38

DATA ELEMENT: Are storage requirements allocated to design?

Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The storage requirements for the system are to be allocated to the individual modules during the Design Phase. This metric is a binary measure of whether or not that allocation is explicitly made.

EXAMPLE: These requirements specify the type of storage (disk, tape) required and the speed of the storage, amount of storage

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfPDM.1

Ef-20

NAME: Use of Virtual Storage

INDEX NUMBER: 39

DATA ELEMENT: Are virtual storage facilities
used?

Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The use of virtual storage or paging techniques enhances the storage efficiency of a system. This metric is a binary measure of whether or not these techniques are planned for and are used.

EXAMPLE: Memory management techniques are used in an operating system. Demand - paged management and segmented memory management which use secondary memory storage are referred to as virtual storage.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfPDM.1

Ef-20

NAME: Global Data

INDEX NUMBER: 42

DATA ELEMENT: On this level, is all global data defined only once?

/Y/N/

On this level, are global variables used as defined globally?

/Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design
(2) Detail Design

DESCRIPTION: Global data elements are to be defined in the same manner by all modules. This metric is a measure of the number of modules in which the global data elements are defined in an inconsistent manner. This metric is used for both the Design and Implementation Phases.

EXAMPLE: If a data element is defined as a global variable, it should not be re-defined anywhere else in the program. For instance, if A1 is defined as a global variable the only reference to A1 should be to that global variable - it should not be redefined as a local variable.

EXPLANATION: This is a binary measure answered by a "Yes or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | CoPDM.3 | Co-23 |
| | CoDDM.2 | Co-33 |
| | MaPDM.1 | Ma-16 |
| | MaDDM.2 | Ma-24 |

NAME: Data Efficiency

INDEX NUMBER: 43

DATA ELEMENT: On this level, have the data base or the files been organized for efficient processing? Y / N

On this level, is the data indexed or referenced efficiently? Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design
(2) Detail Design

DESCRIPTION: Both the data organization and the linkage scheme between data items affects the efficiency of processing. This metric is a binary measure of whether or not the indexing utilized for the data was chosen to facilitate processing.

EXAMPLE:

- 1) Have the number of record types been kept to a minimum?
- 2) Have all the fields within a record that are frequently accessed together physically close together?
- 3) Data bases should be kept as small as possible.
- 4) Data structures should be decided on to make accessing the data base easy (i.e. records, lists, trees).
- 5) Buffers and arrays should be kept as small as possible.

EXPLANATION:: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | EfPDM.2 | Ef-21 |
| | EfDDM.1 | Ef-28 |

NAME: Data Packing

INDEX NUMBER: 44

DATA ELEMENT: On this level, is data packing used?

Y / N

On this level, is bit/byte packing/unpacking kept out of loops?

LIFE CYCLE PHASE(S): (1) Preliminary Design
(2) Detail Design

DESCRIPTION: This metric is a binary measure indicating the overhead involved in packing/unpacking. Placing these activities within loops should be avoided if possible.

EXAMPLE: (*MAIN*)

```
BEGIN  
REPEAT  
  READ INPUT(INPUT)  
UNTIL INPUT(i) = "C";  
END.
```

PROCEDURE READINPUT(VAR input);

Type

Input : packedarray[1..20] of char;

Var

Readin : Input;

Readout : Input;

BEGIN

(i:=1)

Unpack (Readin)

If Readin [i] 'C' then

.

.

Else

Pack (Readin);

END; (*Readinput*)

System: are commands packed for passing?
Module: are arrays packed for storage and then
unpacked when processing occurs? Packing should
not be done within loops because the overhead of
packing & unpacking overrides the benefits

EXPLANATION:

The above code contains an example of unpack-
ing/packing an array where it is unnecessary and
inefficient.

This is a binary measure answered by a "Yes or a
"No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfPDM.3
EfDDM.3

Ef-22
Ef-30

NAME: Communication Protocol Standards

INDEX NUMBER: 46

DATA ELEMENT: Have protocol standards been established and are they followed? / Y / N /

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The communication protocol standards for communication with other systems are to be established during the Design Phase and followed during the Implementation Phase. This binary measure is applied during both of these phases and indicates whether or not the standards are established and followed.

EXAMPLE: (1) Standard file format.
(2) Standard error handling protocol.
should be established (this list is not exhaustive).

EXPLANATION: This is a binary measure answered by a "Yes or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | IpPDM.1 | Ip-22 |

NAME: Module Interface for Input

INDEX NUMBER: 47

DATA ELEMENT: Number of modules used for input from other systems.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The more modules there are that handle input, the more difficult it is to interface with another system and to implement standard protocols. This measure is based on the inverse of the number of modules which handle input. It is to be applied to the design specification and the source code.

EXAMPLE: Input into a system should be handled by a particular module and distributed accordingly rather than having various modules accept their own input.

EXPLANATION:

WORKSHEET REFERENCE:

FORM CODE

PAGE

IpPDM.1

Ip-22

NAME: Translation Standards

INDEX NUMBER: 49

DATA ELEMENT: Have standard data representations or translation standards between representations been established and are they being complied with? Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: More than one translation from the standard data representations used for interfacing with other systems may exist within a system. Standards for these translations are to be established and followed. This binary measure identifies if the standards are established during the Design Phase and are followed during the Implementation Phase.

EXAMPLE: If an EBCDIC file is to be translated to a 6 bit ASCII file, standards for translation of non-equivalent codes should be set.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | IpPDM.2 | Ip-23 |

NAME: Translation Performance

INDEX NUMBER: 50

DATA ELEMENT: Number of modules used to perform translations.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The more modules there are that perform the translation, the more difficult it is to interface with other systems and to implement standard protocols. This measure is based on the inverse of the number of modules which performs a translation.

EXAMPLE: One general purpose translation routine should exist which can translate system-foreign input data files on tape.

EXPLANATION: This metric is scored by dividing one by the number of modules used to perform translations

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | IpPDM.2 | Ip-23 |

NAME: Hard Copy Maintenance

INDEX NUMBER: 53

DATA ELEMENT: Is a hard copy of operator interactions to be maintained?

/Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This is a capability that must be planned for in the Design Phase and coded during the Implementation Phase. It assists in correcting operational errors and improving the efficiency of operation. This metric is a binary measure of whether or not hard copy maintenance is considered in the Design and Implementation Phases.

EXAMPLE: Is a printer copy of operator commands/responses kept? For instance, a "DAYFILE" consisting of all interactions.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.1

Us-24

NAME: Documentation of Lesson Plans/Training Material

INDEX NUMBER: 54

DATA ELEMENT: Are lesson plans/training materials provided for operators, users, and maintainers? /Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This metric is a binary measure of whether or not lesson plans/training materials are considered in the Design and Implementation Phases.

EXAMPLE: Trial and error methods for using a software system can be costly. Is there material to aid in the training of users?

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.4 | Us-18 |

NAME: Realistic Simulated Exercises

INDEX NUMBER: 55

DATA ELEMENT: Are realistic, simulated exercises provided?

Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This metric is a binary measure of whether or not exercises which represent the operation environment are developed during the Implementation Phase for use in training.

EXAMPLE: It can be costly and time consuming learning to use a software system using "live" data. A simulation of the system should be provided using dummy data to train users on the system.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsRAM.4

Us-18

NAME: Sufficient "Help" and Diagnostic Information

INDEX NUMBER: 56

DATA ELEMENT: Are "Help" and diagnostic information provided? Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This metric is a binary measure of whether or not the capability to aid the operator in familiarization with the system has been designated and built into the system.

EXAMPLE: A list of legal commands or a list of the sequential steps involved in a process are examples of this metric.

*"Enter Command Prompt"
(Type "Help" for Available Commands)

*Help - (Return)

The System Prompts With
PRT - Prints Out Quarterly Statistics
UPD - Updates Quarterly Statistics
DEL - Deletes Quarterly Statistics

*"Enter Command Prompt"

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsRAM.4 | Us-18 |

NAME: Uniform Input Formats

INDEX NUMBER: 57

DATA ELEMENT: Number of input formats.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The greater the number of input formats there are, the more difficult it is to use the system. This measure is based on the total number of input formats.

EXAMPLE: The various ways a date can be input will serve as an example. When the system is started the date may be input as 03-01-81. When reports are generated the input format may be 030181 and another module may request the date in the format 03/01/81. These are three different input formats for the same data and they make the system harder to use.

EXPLANATION: This metric is scored by dividing one by the number of different input record formats.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsPDM.2 | Us-26 |

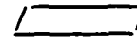
NAME: Default Values

INDEX NUMBER: 58

DATA ELEMENT: Number of default values.



Number of input Parameters.



LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: A method of minimizing the amount of input required is to provide defaults. This measure is based on the number of defaults allowed, divided by the total number of input parameters. It is applied during the Design and Implementation Phases.

EXAMPLE: Commands that are frequently executed could be given default values in order to minimize typing errors.

if this command is frequently executed:

COPY fileoriginal=INPUT filecopy=OUTPUT records=ALL s=0

default values of INPUT, OUTPUT, ALL and 0 should be given to this command so that it is only necessary to type:

COPY

to execute the same command.

EXPLANATION: The metric is scored by dividing the number of default values by the number of input parameters.

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.2

Us-26

NAME: Self-identifying Inputs

INDEX NUMBER: 59

DATA ELEMENT: Number of self-identifying input values. /____/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: Input records which have self-identifying codes enhance the accuracy of user inputs. This measure is based on the number of input records. It is applied during the Design and Implementation Phases.

EXAMPLE: Input records consist of:
/1234/NAME/ADDRESS/PHONE/ Record - Type 1
/5678/NAME/S.S. NO/SALARY/ Record - Type 2
/9ABC/ADDRESS/BENEFITS/VACATION/ Record - Type 3

EXPLANATION: (1) The first field is a code that identifies the type of record that is being sent so that the receiving module needs only decide which record - type is present in order to process it.

(2) This metric is scored by dividing the number of self identifying input records by the total number of input records.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsPDM.2 | Us-26 |

NAME: Input Verification

INDEX NUMBER: 60

DATA ELEMENT: Can input be verified by the user
prior to execution?

Y/N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The capability to display input on request or to echo input automatically enables the user to check his inputs before processing. This metric is a binary measure of whether or not this capability is provided in the Design and Implementation Phases.

EXAMPLE: This data element refers to the usability of the system. The user should be able to see the input before a program executes, so that bad input can be identified.

EXPLANATION: This is a binary measure answered by a "Yes or a "No."

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.2

Us-27

NAME: Explicitly Defined Logical End of Input

INDEX NUMBER: 61

DATA ELEMENT: Is input terminated by explicitly defining the end of input? /Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The user should not have to provide a count of input cards. This metric is a binary measure of whether or not this capability is provided in the Design and Implementation Phases.

EXAMPLE: An end-of-data card may be used to determine the end of input.

EXPLANATION: This is a binary measure answered by a "Yes or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsPDM.2 | Us-27 |

NAME: Techniques for Separating Logical Groups of Output

INDEX NUMBER: 62

DATA ELEMENT: Are logical groups of output separated for user examination? Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: Utilization of top of page, blank lines, lines of asterisks provide for easy identification of logically grouped output. This metric is a binary measure of whether or not these techniques are used during the Design and Implementation Phases.

EXAMPLE: Separation of a month of daily reports by blank lines or asterics is an example.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".:

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | UsPDM.3 | Us-28 |

NAME: Relationship Between Error Messages and Outputs

INDEX NUMBER: 63

DATA ELEMENT: Are the relationships between error messages and outputs unambiguous?

Y / N /

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This metric is a binary measure of whether or not error messages will be directly related to the output. It is used during the Design and Implementation Phases.

EXAMPLE: For each output file error messages should be found in the outputs for that file. The error message should be relevant and unambiguous: i.e., line numbers identifying the source of error and some type of diagnostic message should be present.

"ERROR 295" is not diagnostic.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.3

Us-28

NAME: Path Coverage

INDEX NUMBER: 64

DATA ELEMENT: Number of paths to be tested. /

Total number of paths. /

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: Plans for testing the various paths within a module should be made during the Design Phase and the test cases actually developed during the Implementation Phase. This measure identifies the number of paths planned to be tested, divided by the total number of paths.

EXAMPLE: The paths of a module usually begin at the level of logic branches. Each logic branch in a module causes 2 or more paths to begin. An example of a logic branch is a statement such as a GOTO or an IF. Each pathway should be tested to determine that the logic is correct.

EXPLANATION: This metric is scored by dividing the number of paths to be tested by the total number of paths.

WORKSHEET REFERENCE:

FORM CODE

PAGE

TePDM.2

Te-17

NAME: Input Parameters Boundary Tested

INDEX NUMBER: 65

DATA ELEMENT: Number of input parameters to be tested.

Total number of input parameters.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: Another aspect of module testing involves testing the input ranges to the module. This is done by exercising the module at the various boundary values of the input parameters. Plans to do this must be specified during the Design Phase and coded during the Implementation Phase. The measure is the number of parameters to be boundary tested, divided by the total number of parameters.

EXAMPLE: Input Parameter: Input values can range from 0 to 9999. If the low boundary (0) and the high boundary (9999) are not tested for accuracy of logic and computation - then the program has not been tested fully.

EXPLANATION: This metric is scored by dividing the number of parameters to be boundary tested by the total number of parameters.

WORKSHEET REFERENCE:

FORM CODE

PAGE

TePDM.2

Te-17

NAME: Module Interfaces Tested

INDEX NUMBER: 66

DATA ELEMENT: Number of interfaces to be tested.

Total number of interfaces.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: One aspect of integration testing is the testing of all module-to-module interfaces. Plans to accomplish this testing are prepared during the Design Phase and the tests are developed during the Implementation Phase. The measure is based on the number of interfaces to be tested divided by the total number of interfaces.

EXAMPLE: Below is a sample statement of interface testing requirements for a C³ system:

"The qualification of the requirement for the functions of the application software to interface with each other through executive linkage calls to the Operating System (OS) with parameters passed through the inter-program packet (IPP) and the qualification of the requirement for the functions of the applications software to interface with the OS will be accomplished by FQT implicit and explicit testing on the PODM by using satisfactory qualification of the CNCE software functional requirements (See Section 4.2); by visual examination of detailed listings; by observation of man/machine interfaces, controller messages, and/or error notifications; and by examination of memory maps, core dumps, disk dumps, or directory listings."

EXPLANATION: This metric is scored by dividing the number of module interfaces to be tested the total number of module interfaces.

WORKSHEET REFERENCE:

FORM CODE

PAGE

TePDM.3

Te-18

NAME: Performance Requirements Coverage

INDEX NUMBER: 67

DATA ELEMENT: Number of performance requirements
to be tested.

Total number of performance
requirements.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The second aspect of integration testing involves checking for compliance with the performance requirements at the module and subsystem level. This testing is planned during the Design Phase and the tests are developed during the Implementation Phase. The measure is the number of performance requirements to be tested, divided by the total number of performance requirements.

EXAMPLE: A separate performance requirements matrix (See 29) could be compiled to check the coverage of the performance requirements (such as run time requirements, storage and response time requirements) for each module. This would make it easy to determine if all modules have been covered.

EXPLANATION: This metric is scored by dividing the total number of performance requirements to be tested by the total number of performance requirements.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | TePDM..3 | Te-18 |

NAME: Module Coverage

INDEX NUMBER: 68

DATA ELEMENT: Number of modules to be exercised.

Number of modules.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: One aspect of system testing which can be measured as early as the Design Phase is the equivalent to path coverage at the module level. For all system test scenarios planned, the percent of all of the modules to be exercised is important.

EXAMPLE: Each module itself must be tested, and also all groups of modules that interact must be tested together to test the integration.

EXPLANATION: This metric is scored by dividing the number of modules to be exercised by the total number of modules.

WORKSHEET REFERENCE:

FORM CODE

PAGE

TePDM.4

Te-19

NAME: Identification of Test Inputs and Outputs in Summary Form

INDEX NUMBER: 69

DATA ELEMENT: Are test inputs and outputs provided in summary form? /Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The results of tests and the manner in which these results are displayed are very important to the effectiveness of testing. This is especially true during system testing because of the potentially large volume of input and output data. This measure simply identifies if the capability exists to display test inputs and outputs in a summary fashion. The measure can be applied to the plans and specifications in the Design Phase and the development of this capability during the Implementation Phase.

EXAMPLE: Are there plans for summarizing the testing reports and descriptions of the formats for these reports. This description might include:

Format description
Content description
Time schedule

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | TePDM.4 | Te-19 |

NAME: Size of Data Base

INDEX NUMBER: 70

DATA ELEMENT: Number of unique data items in data base.

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: The size of the data base, in terms of the number of unique data items contained in the data base, relates to the design structure of the software system. A data item is a unique data element, as for example, an individual data entry or data field.

EXAMPLE: The number of separate fields that compose a record is one way of measuring uniqueness.

EXPLANATION: This metric is scored by dividing one by the number of unique data items.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | RePDM.4 | Re-28 |
| | MaPDM.2 | Ma-17 |
| | TePDM.1 | Te-16 |

NAME: Detected Error Condition

INDEX NUMBER: 72

DATA ELEMENT: When an error condition is detected, Y / N
is it passed to the calling module?

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: The decision of what to do about an error is to be made at a level where an affected module is controlled. This concept is built into the design and then implemented.

EXAMPLE: When an error occurs in a subroutine, an error message should be returned to the calling program, and the calling program should determine further action: halting, correcting mistake and continuing, bypassing the problem.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.1 | Re-34 |
| | ReIMM.1 | Re-47 |

NAME: Sufficiency of Numerical Methods

INDEX NUMBER: 73

DATA ELEMENT: Have the numerical techniques being used in an algorithm been analyzed with regards to accuracy requirements?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: The numerical methods utilized within the system are to be consistent with the accuracy objectives. They can be checked in the Design and Implementation Phases.

EXAMPLE: For values whose tolerances are critical, have the most accurate rounding techniques been performed at the most appropriate time in the formulas (errors become cumulative)?

EXPLANATION: This is a binary measure answered by as "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReDDM.4

Re-37

NAME: Value of Input Ranges

INDEX NUMBER: 74

DATA ELEMENT: Are values of inputs range tested?

Y / N

LIFE CYCLE PHASE(S):
(1) Detail Design
(2) Implementation

DESCRIPTION: The attributes of each input item are to be checked for reasonableness. These checks are to be specified in the Design Phase and to exist in code in the Implementation Phase.

EXAMPLE: Inputs must be checked to verify that they are numeric, alphabetic, positive, or negative of a certain length, or nonzero, greater than or less than a certain value, etc.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.2 | Re-35 |
| | ReIMM.2 | Re-48 |

NAME: Redundant Input Data

INDEX NUMBER: 75

DATA ELEMENT: Are conflicting requests and illegal combinations identified and checked? Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: Checks (to see if redundant input data agrees, if combinations of parameters are reasonable, and if requests are conflicting) should be documented in the Design Phase. These checks should exist in the code in the Implementation Phase.

EXAMPLE: Copy Input, Output S=a R=b

EXPLANATION: If the values allowed by S and R can only be numeric, but the input is alphanumeric, an error will occur. Checks for agreement between value ranges, etc, are necessary and should be included at these phases.
This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.2 | Re-35 |
| | ReIMM.2 | Re-48 |

NAME: Sufficiency of Input Data

INDEX NUMBER: 76

DATA ELEMENT: Is there a check to see if all
necessary data is available before
processing begins?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: To avoid going through several processing steps
before incomplete input data is discovered,
checks for sufficiency of input data are to be
made prior to the start of processing.

EXAMPLE: Open input data files at the start of process-
ing. If a file does not exist or contains
erroneous data, it will be discovered before
processing begins.

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReDDM.2
ReIMM.2

Re-35
Re-48

NAME: Input Checking

INDEX NUMBER: 77

DATA ELEMENT: Is all input checked, reporting all errors, before processing begins?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: Input checking is not to stop at the first error encountered, but is to continue through all the input and then to report all errors. Processing is not to start until the errors are reported and corrections are made or a continue processing command is given.

EXAMPLE: A tape processor checking for errors is an example of input checking. A tape processor reads an entire tape of input, searching for serious errors which can be eliminated before processing begins.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.2 | Re-35 |
| | ReIMM.2 | Re-48 |

NAME: Range Test for Loop and Multiple Transfer Index Parameters

INDEX NUMBER: 78

DATA ELEMENT: Are loop and multiple transfer index parameters range tested before use? Y/N

LIFE CYCLE PHASE(S): (1) Detail Design.
(2) Implementation

DESCRIPTION: Range tests for loop indices and multiple transfers are to be specified in the Design Phase. These range tests are to exist in code in the Implementation Phase.

EXAMPLE: Loop Indices
DIMENSION A(120), B(10)
DO 100 I=T,N
A(I) = 0
100 CONTINUE

Multiple Transfers
GOTO(3,57,100,4),P

You must range test for T, N and P here. If T=0 there will be problems and if N is greater than 120 it may write over into ARRAY B, and if P is less than 1 or greater than 4, there will be problems.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.3 | Re-36 |
| | ReIMM.3 | Re-49 |

NAME: Subscript Checking

INDEX NUMBER: 79

DATA ELEMENT: Are subscript value ranges tested
before use?

Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: Checks for legal subscript values are to be
specified in the Design Phase and coded during
the Implementation Phase.

EXAMPLE: DIMENSION COST (12,70)

Any reference to the above array must have legal
subscript values. If you use the statement
cost (year, loc), then year must be 12 or less
but not zero and loc should be 70 or less but
not zero.

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReDDM.3

Re-36

ReIMM.3

Re-49

NAME: Output Checking

INDEX NUMBER: 80

DATA ELEMENT: Is output checked for reasonableness before processing continues?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: Certain range-of-value checks are to be made during processing to ensure the reasonableness of final outputs. This is usually done only for critical parameters. These are to be identified during the Design Phase and coded during the Implementation Phase.

EXAMPLE: The results from evaluating critical formulas should be checked for reasonableness. If a critical formula determines "altitude", a check should be made to make sure that the altitude is positive.

EXPLANATION: This a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.3 | Re-36 |
| | ReIMM.3 | Re-49 |

NAME: Data and Control Flow Complexity

INDEX NUMBER: 81

DATA ELEMENT: What is the sum of the number of
decision points, subdecision points,
conditional branches, and uncondi-
tional branches?

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: This metric can be measured from the design representation (e.g., flow charts) and the code automatically. Path flow analysis and variable set/use information along each path is utilized. A variable is considered to be 'live' at a code if it can be used again along that path in the program. The complexity measure is based on summing the 'liveness' of all variables along all paths in the program. It is normalized by dividing it by the maximum complexity of the program (all variables live along all paths).

EXAMPLE:

C*****
C GENERATE RADAR REPORTS

```
30 CONTINUE
(1) DO 32 T=1,NTGT
    READ(2) XT(T),YT(T),RT,AZT,VT(T),HT(T),RDT
    CALL GAUSS (IX(T+2),SIGAXR,O.O,ERA)
    CALL GAUSS (IX(T+2),SIGR,O.O,ERR)
    CALL GAUSS (IX(T+2),SIGRD,O.O,ERRD)
    CALL RANDU (IX(T+2),IY,RNDM)
    IX(T+2) = Y
(2) IF(N.LT.NI(T).OR.N.GT.NF(T)) GO TO 35
(3) IF (N.EQ.NI(T)) GO TO 40
(4) IF (RNDM.GT.PD.OR.ABS(RDT).LE.80.0) GO TO 35
40 CONTINUE
    RG(T,1) = RT + ERR
    AZ(T,1) = AMDD(AZT/DEG + ERA.2.0*PI)
    RD(T)=RDT+ERRD
```

```

XR(T.1)=RG(T.1*SIN(AZ(T.1)))+XA
YR(T.1)=RG(T.1*COS(AZ9T.1))+TA
ITRACK(T.N) = -1
(5) GO TO 36
35 RG(T.1)=0
   AZ(T.1)=0
   RD(T)=0
   XR(T.1)=0
   YR(T.1)=0

```

C RADAR REPORTS GENERATED
C*****

EXPLANATION: There is a total of 5 decision points, subdecision points, conditional and unconditional branches.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.5 | Re-38 |
| | ReIMM.6 | Re-52 |
| | MaDDM.4 | Ma-26 |
| | MaIMM.2 | Ma-37 |
| | TeDDM.1 | Te-28 |
| | TeIMM.2 | Te-38 |

NAME: Module Processing not Dependent on Prior Processing

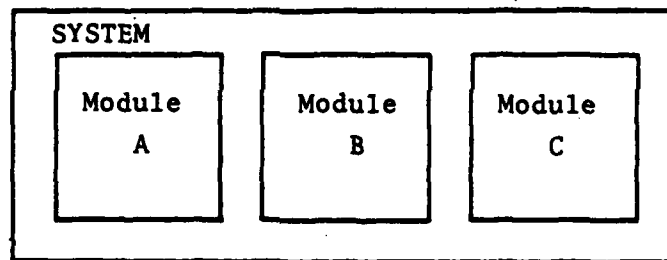
INDEX NUMBER: 82

DATA ELEMENT: Is the module independent of knowledge of prior processing? Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: The processing done within a module is not to be dependent upon knowledge of or results of prior processing, e.g., the first time through the module, the 9th time through. This rule is applied in both the Design and Implementation Phases.

EXAMPLE:



EXPLANATION: Processing of information thru each module should be independent of the processing of any other module; changes made to one module should not require changes in any other module.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.6 | Re-39 |
| | MaDDM.3 | Ma-25 |
| | TeDDM.2 | Te-29 |

NAME: Entrance and Exit of the Module

INDEX NUMBER: 83

DATA ELEMENT: Number of entrances into modules.

Number of exits from module.

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: Each module has single entrance, single exit. Determination of the number of modules that violate this rule at design and implementation can be made and is the basis for the metric.

EXAMPLE:

Example of Multiple Entrances

```
PROGRAM DODO
CALL ENT1
CALL ENT2
STOP
END

IDENT MULTENT
ENTRY ENT1
ENTRY ENT2
ENT1 BSS 1
EQ ENT1
ENT2 BSS 1
EQ ENT2
END
```

Example of Multiple Exits

```
FUNCTION RATE(Q)
RATE=1.0
IF (Q.GT.3.0)GOTO 3
RETURN (--- one exit
3 IF (Q.GT.5.0)GOTO 7
RATE=2.0
RETURN (--- 2nd exit
```

7 RATE=3.0
RETURN
END

(--- 3rd exit

EXPLANATION:

The first example is a combination of COMPASS and FORTRAN. There are two entrances to the COMPASS routine "MULTENT", and they are entered via the FORTRAN program "DODO". The multiple exit function contains 3 exits (marked with arrows).

This metric is scored by dividing 1 by the number of entrances plus 1 divided by the number of exits.

WORKSHEET REFERENCE:

FORM CODE

PAGE

| | |
|---------|-------|
| ReDDM.6 | Re-39 |
| ReIMM.5 | Re-51 |
| MaDDM.3 | Ma-25 |
| MaIMM.1 | Ma-36 |
| TeDDM.2 | Te-29 |

NAME: Accuracy, Convergence, and Timing Attributes

INDEX NUMBER: 84

DATA ELEMENT: Are accuracy, convergence, or timing attributes parametric? / Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: A module which can provide varying degrees of convergence or timing to achieve greater precision provides the attribute of extensibility. Hard-coded control parameters, counters, and clock values violate this measure. This measure is based on the number of modules which do not exemplify this characteristic. A determination can be made during the Design and Implementation Phases.

EXAMPLE: A scientific program contains a function which evaluates a formula containing variable degrees of accuracy requirements. The accuracy is determined by a parameter passed to the routine (=3.14 or =3.1415, etc).

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | FxDDM.2 | Fx-16 |
| | FxIMM.3 | Fx-29 |

NAME: Dependence on Software System Utility Programs

INDEX NUMBER: 85

DATA ELEMENT: Number of references to system library routines, utilities, or other system provided facilities or functions.

Number of total lines of code.

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: The more utility programs, library routines, and other system facilities that are used within a system, the more dependent the system is on that software system environment.

EXAMPLE: Search subroutines & tables lookups, IO routines for minicomputers, Sorts & Merges, Dumps, Traces, Snapshots, Debugging Packages, etc., Conversion, Subroutines, Simple IO Utility Subroutines (Tape-to-Printer, Card-to-Disk, etc.) & Dynamic Storage Allocation Subroutines in one operating system are unlikely to be exactly similar to other operating systems.

EXPLANATION: This measure is based on the number of references to system facilities in a module divided by the total number of lines of code in the module.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | PoDDM.3 | Po-18 |
| | PoIMM.6 | Po-32 |
| | RuDDM.4 | Ru-19 |
| | RuIMM.6 | Ru-36 |

NAME: Controlling Parameters

INDEX NUMBER: 87

DATA ELEMENT: Number of calling sequence parameters that are control variables.

Number of calling sequence parameters.

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: The calling module defines the controlling parameters, any input data required, and the output data required. Control must also be returned to the calling module. This measure is based on the number of calling parameters which are control parameters. They can be measured in the Design and Implementation Phases.

EXAMPLE:

MAIN PROGRAM

```
Dimension A(33), B(33), C(33)
Dimension T(22), Q(22), R(22)
M=3
N=3
Call Add (A, B, C, M, N)
```

SUBPROGRAM

```
Subroutine Add (X,Y,Z,M,N)
Dimension X(M,N), Z(M,N)
Do 10 I=1,M
Do 10 J=1,N
10 Z(I,J)=X(I,J)+Y(I,J)
Return
```

EXPLANATION: M,N are control variables. A,B,C are dummy variables because they only pass data. M,N are control variables because they control the actions of the called subroutines. Total number of calling sequence parameters is 5, number of control variables is 2.

WORKSHEET REFERENCE:FORM CODEPAGE

| | |
|---------|-------|
| MaDDM.5 | Ma-27 |
| MaIMM.4 | Ma-41 |
| FxDDM.3 | Fx-17 |
| FxIMM.1 | Fx-27 |
| TeDDM.3 | Te-30 |
| TeIMM.4 | Te-42 |
| PoDDM.1 | Po-15 |
| PoIMM.1 | Po-25 |
| RuDDM.1 | Ru-15 |
| RuIMM.1 | Ru-28 |
| IpDDM.1 | Ip-29 |
| IpIMM.1 | Ip-35 |

NAME: Controlling Input

INDEX NUMBER: 88

DATA ELEMENT: Is input passed as calling sequence parameters?

Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: See "Controlling Parameters 87"

EXAMPLE:

MAIN PROGRAM

Dimension A(100)
Call Readrecord (A)

SUBROUTINE

Subroutine Readrecord (A)
Dimension A(100)

EXPLANATION: The input record was passed as an array to the subroutine read record. The answer is "Yes" in this case.

This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

| | |
|---------|-------|
| MaDDM.5 | Ma-27 |
| MaIMM.4 | Ma-41 |
| FxDDM.3 | Fx-17 |
| FxIMM.1 | Fx-27 |
| TeDDM.3 | Te-30 |
| TeIMM.4 | Te-42 |
| PoDDM.1 | Po-15 |
| PoIMM.1 | Po-25 |
| RuDDM.1 | Ru-15 |
| RuIMM.1 | Ru-28 |
| IpDDM.1 | Ip-29 |
| IpIMM.1 | Ip-35 |

NAME: Controlling Output

INDEX NUMBER: 89

DATA ELEMENT: Is output data passed back to
calling module?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: See "Controlling Parameters 87"

EXAMPLE: Output data from the subroutine should be passed
back to the calling program in parameters as
opposed to using COMMONS.

EXPLANATION: This is a binary measure answered by a "Yes" or
a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaDDM.5 | Ma-27 |
| | MaIMM.4 | Ma-41 |
| | FxDDM.3 | Fx-17 |
| | FxIMM.1 | Fx-27 |
| | TeDDM.3 | Te-30 |
| | TeIMM.4 | Te-42 |
| | PoDDM.1 | Po-15 |
| | PoIMM.1 | Po-25 |
| | RuDDM.1 | Ru-15 |
| | RuIMM.1 | Ru-28 |
| | IpDDM.1 | Ip-29 |
| | IpIMM.1 | Ip-35 |

NAME: Controlling Return

INDEX NUMBER: 90

DATA ELEMENT: Is control returned to calling module?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: See "Controlling Parameters 87"

EXAMPLE: Processing should never branch to a subroutine and end there. Control should always be returned to calling module.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaDDM.5 | Ma-27 |
| | MaIMM.4 | Ma-41 |
| | FxDDM.3 | Fx-17 |
| | FxIMM.1 | Fx-27 |
| | TeDDM.3 | Te-30 |
| | TeIMM.4 | Te-42 |
| | PoDDM.1 | Po-15 |
| | PoIMM.1 | Po-25 |
| | RuDDM.1 | Ru-15 |
| | RuIMM.1 | Ru-29 |
| | IpDDM.1 | Ip-29 |
| | IpIMM.1 | Ip-35 |

NAME: Share Temporary Storage

INDEX NUMBER: 91

DATA ELEMENT: Is temporary storage independent of other modules? Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: This is a binary measure to determine whether or not modules share temporary storage. It emphasizes the loss of module independence if temporary storage is shared between modules.

EXAMPLE: Storage should be separate for each module. Accessing commons should not be used instead of passing parameters from one routine to the next.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaDDM.5 | Ma-28 |
| | FxDDM.3 | Fx-18 |
| | TeDDM.3 | Te-31 |
| | PoDDM.1 | Po-16 |
| | RuDDM.1 | Ru-16 |
| | IpDDM.1 | Ip-30 |

NAME: Unmixed Processing and Input/Output

1. EX NUMBER: 92

DATA ELEMENT: Does the module not mix input,
output, and processing functions
in the same module?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: A module which performs input/output as well as processing is not as general as a module which simply accomplishes the processing. This measure is based on the number of modules that violate this concept in the Design and Implementation Phases.

EXAMPLE: There should be one module whose function it is to read the input, another module should write to output, and another module for each intermediate process that is performed. Mixing of functions should not be done in one module.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | FxDDM.4 | Fx-19 |
| | FxIMM.2 | Fx-28 |
| | RuDDM.2 | Ru-17 |
| | RuIMM.2 | Ru-30 |

NAME: Machine Dependent Functions

INDEX NUMBER: 93

DATA ELEMENT: Number of machine dependent functions performed.

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: Any references to machine dependent functions within a module lessens its generality. This measure is based on the number of machine dependent functions in a module.

EXAMPLE: An example would be referencing the system clock for timing purposes.

EXPLANATION: The more independent the software is, the more general it is.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | FxDDM.4 | Fx-19 |
| | FxIMM.2 | Fx-28 |
| | RuDDM.2 | Ru-17 |
| | RuIMM.2 | Ru-30 |

NAME: Unlimited Data Volume

INDEX NUMBER: 94

DATA ELEMENT: Is processing data volume unlimited? Y / N

Can the amount of data that can be processed be unlimited? Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: A module which has been designed and coded to accept no more than 100 data item inputs for processing is certainly not as general in nature as a module which will accept any volume of input. This measure is based on the number of modules which are designed or implemented to be data volume limited.

EXAMPLE: A file with an unknown number of data items can be read in sequentially. Open the file and read the data elements and place in an array. Use a counter variable "count") to record the number of data items. If you wish to process in a loop for instance, the loop indices could be DO 10 I = 1, count.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | FxDDM..5 | Fx-19 |
| | FxIMM.2 | Fx-28 |
| | RuDDM.2 | Ru-17 |
| | RuIMM.2 | Ru-30 |

NAME: Unlimited Data Value

INDEX NUMBER: 95

DATA ELEMENT: In the Design Phase, is processing data value unlimited? Y / N

In the Implementation Phase, is the value of data that can be processed unlimited? Y / N

LIFE CYCLE PHASE(S): (1) Detail Design
(2) Implementation

DESCRIPTION: A previously identified element, ET.2(2) of Error Tolerance dealt with checking input for reasonableness. This capability is required to prevent providing data to a function for which it is not defined or its degree of precision is not acceptable. This is a necessary capability from an error tolerance viewpoint. From a generality viewpoint, the smaller the subset of all possible inputs to which a function can be applied the less general it is. Thus, this measure is based on the number of modules which are data value limited. This can be determined in the Design and Implementation Phases.

EXAMPLE: Detail Design: the values of the inputs to each function should have as few constraints as possible (i.e., few precision requirements).

Implementation: The values of the input to each module should have as few constraints as possible.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | FxDDM.4 | Fx-19 |
| | FxIMM.2 | Fx-28 |
| | RuDDM.2 | Ru-17 |
| | RuIMM.2 | Ru-30 |

AD-A120 377

SYSTEMS ARCHITECTS INC RANDOLPH MASS

F/G 5/2

COMPUTER SYSTEMS ACQUISITION METRICS HANDBOOK, VOLUME III, DATA--ETC(II)

MAY 82

F19628-B0-C-0207

UNCLASSIFIED

ESD-TR-82-143(3)

NL

20 2

0000

END
DATE
FILMED
11 82
DTF

NAME: Standard Language Used

INDEX NUMBER: 96

DATA ELEMENT: Is a common standard subset of a programming language to be used?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: The use of nonstandard constructs of a language that may be available from certain compilers can cause conversion problems when the software is moved to a new software system environment. This measure represents that situation. It is based on the number of modules which are coded in a nonstandard subset of the language. The standard subset of the language is to be established during design and adhered to during the Implementation Phase.

EXAMPLE: Mixed mode operations, negative indices in a do loop, are examples of non-standard constructs of a language.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | PoDDM.3 | Po-18 |
| | RuDDM.4 | Ru-19 |

NAME: Programming Language Availability

INDEX NUMBER: 97

DATA ELEMENT: Is the programming language available in other machines?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: This is a binary measure which identifies whether or not the programming language used is available on other machines. This refers to the same version and dialect of the language.

EXAMPLE: One language may have different versions e.g., Fortran IV and Fortran 77. Assembly language also tends to be machine dependent. A DEC assembly language cannot be used on a IBM machine

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

PoDDM.2
RuDDM.3

Po-17
Ru-18

NAME: Logical Processing Independence

INDEX NUMBER: 98

DATA ELEMENT: Is logical processing independent
of storage specification?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: The logical processing of a module is to be independent of storage size, buffer space, or array size. The design provides for variable dimensions and dynamic array sizes to be defined parametrically. The metric is based on the number of modules containing hard coded dimensions which do not exemplify this concept.

EXAMPLE:

MAIN PROGRAM

Dimension A(100)
C=100
.
.

SUBROUTINE

Subroutine Process Rec (A,C)
Dimension A(C)
.
.

RETURN

EXPLANATION: The size of the array in the subroutine does not depend on knowledge of array size. It receives its value thru a parameter passed to it and thus can be modified easily.

This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

FxDDM.1

Fx-15

NAME: Modules Table Driven

INDEX NUMBER: 99

DATA ELEMENT: Is the module table driven?

Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: The use of tables within a module facilitates different representations and processing characteristics. This measure is based on the number of modules which are not table driven, and can be applied in the Design and Implementation Phases.

EXAMPLE: A system contains a general-purpose translation program which will read a table to translate the input data. The program determines the character, and references its position in the Table and substitutes the translated character. There are no IF or CASE statements needed.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

FxDDM.2

Fx-16

NAME: Non-Loop Dependent Computations Kept Out of Loop

INDEX NUMBER: 100

DATA ELEMENT: How many non-loop functions are kept out of loops?

Total number of loop statements:

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: Such practices as evaluating constants in a loop are to be avoided. This measure is based on the number of non-loop dependent statements found in all loops in a module. This is to be measured from a detail design representation during the Design Phase and from the source code during the Implementation Phase.

EXAMPLE: GOOD (Q is evaluated before the loop begins)

```
Q=(DUMP+BUMP+GRIND) (--- Non-loop dependent
DO 20 1,4,2
  IF (Q.EQ.MAYBE) GOTO 35
  POP(Z)
  IF (Q.EQ.DEFNO) GOTO 40
  POP(K)
20 CONTINUE
```

BAD (The formula (DUMP+BUMP+GRIND) should be evaluated before entering the loop)

```
DO 20 1,4,2
  IF ((DUMP+BUMP+GRIND).EQ.MAYBE) GOTO 35
  POP(Z)
  IF ((DUMP+BUMP+GRIND).EQ.DEFNO) GOTO 40
  POP(K)
20 CONTINUE
```

EXPLANATION: This metric is scored by dividing the number of non-loop dependent statements in loop by the total number of loop statements.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | BfDDM.3 | Bf-30 |

NAME: Standard Design Representation

INDEX NUMBER: 101

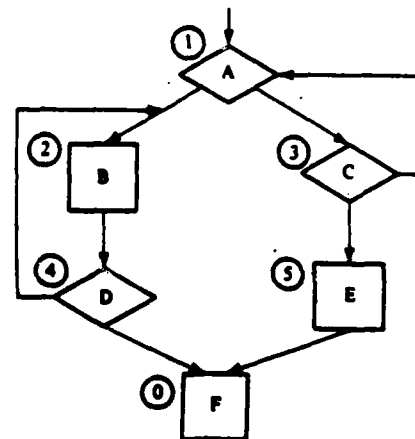
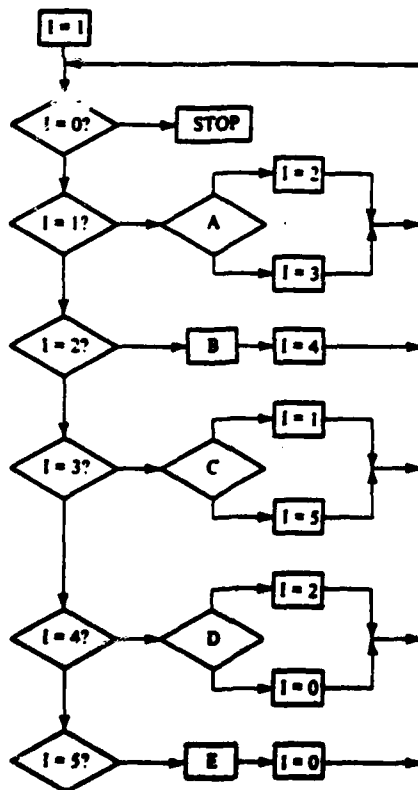
DATA ELEMENT: Does the design representation
comply with established standards?

Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: Flowcharts, HIPO charts, Program Design Languages, or other design representation may be used. Standards for representing the elements of the control flow are to be established and followed. This element applies to the Design Phase only. The measure is based on the number of modules whose design representation does not comply with the standards.

EXAMPLE:



EXPLANATION:

This is a binary measure answered by a "Yes" or a "No". These two designs do not match, so the response would be "no", if they were used on the same system.

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoDDM.1
MaDDM.1

Co-32
Ma-23

NAME: Input/Output Conventions

INDEX NUMBER: 102

DATA ELEMENT: Do input/output references comply
with established standards?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: Conventions for which modules will perform input/output, how it will be accomplished, and the input/output formats are to be established and followed. This measure is based on the number of modules which do not comply with the conventions.

EXAMPLE: Check the source code: If an output format is supposed to conform to: (T4, 'Interest', 2X, I4) and you find (T4, 'Interest', I4), that module does not conform.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoDDM.1
MaDDM.1

Co-32
Ma-23

NAME: Calling Sequence Conventions

INDEX NUMBER: 103

DATA ELEMENT: Do calling sequences comply with established standards?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: Interactions between modules are to be standardized during the Design Phase and followed in the Implementation Phase. This measure is based on the number of modules which do not comply with the conventions.

EXAMPLE: Module A-(Input, Output, Parm1, Parm2 . . Parm N)
Module B-(Output, Input, Parm1)

EXPLANATION: Module B does not follow the convention of input first, output second, and parms last. It is an error.

This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoDDM.1
MaDDM.1

Co-32
Ma-23

NAME: Error Handling Conventions

INDEX NUMBER: 104

DATA ELEMENT: Is error handling done according to established standards?

Y / N

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: A consistent method for error handling is required. Conventions established in the Design Phase are followed in the Implementation Phase. This measure is based on the number of modules which do not comply with the conventions.

EXAMPLE: Conventions established in the analysis phase should be followed when errors occur:

EXAMPLE

- 1) Trap Error
- 2) Affix Code
- 3) Write error message on output file in Format 6.

EXPLANATION: If an error occurs, this format should be followed, if not: this is a reason to answer "no".

This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoDDM.1
MaDDM.1

Co-32
Ma-23

NAME: Naming Conventions

INDEX NUMBER: 105

DATA ELEMENT: Are variables named according
to established standards?

Y / N /

LIFE CYCLE PHASE(S): (1) Detail Design

DESCRIPTION: Naming conventions for variables and modules are
to be established and followed.

EXAMPLE: Analysis Phase: 1st 3 letters are SAI, followed
by department (3 letters) followed by variable
name (3). SAISQAMET = Filename

EXPLANATION: This is a binary measure answered by a "Yes" or
a "no".

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoDDM.2
MaDDM.2

Co-33
Ma-24

NAME: Negative Boolean or Compound Boolean Expressions

INDEX NUMBER: 106

DATA ELEMENT: Number of negative or complicated compound Boolean expressions.

Number of lines excluding comments.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Compound expressions involving two or more Boolean operations and negation can be avoided. These types of expressions add to the complexity of the module. The measure is based on the number of these complicated expressions per executable statement in the module.

EXAMPLE: A COMPOUND BOOLEAN OPERATION would be A .EQ. B
.OR. C .NE. D .AND. E .GT. F
A negative Boolean Expression would be
A .EQ. B .OR. .NOT. C .EQ. D .AND. E .GT. F

EXPLANATION: This metric is measured with the following:
One minus the number of compound or negative Boolean expressions, divided by the number of executable statements.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReIMM.7 | Re-53 |
| | MaIMM.3 | Ma-38 |
| | TeIMM.3 | Te-39 |

NAME: High Order Language

INDEX NUMBER: 107

DATA ELEMENT: Is high order language used?

Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: An HOL is much more self-descriptive than assembly language. The measure is based on the modules which are implemented, in whole or in part, in assembly or machine language.

EXAMPLE: If a module has assembly languages subroutines or a main module is written in assembly language then the response to this data element is "No".

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaIMM.7 | Ma-45 |
| | FxIMM.6 | Fx-33 |
| | TeIMM.7 | Te-46 |
| | PoIMM.4 | Po-29 |
| | RuIMM.5 | Ru-34 |

NAME: Statement Labels

INDEX NUMBER: 108

DATA ELEMENT: (1) Number of statement labels?
(Do not count format statements)
(2) Number of lines excluding comments
(Executable statements)

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The measure is based on the premise that as more statement labels are added to a module, the more complex it becomes to understand.

EXAMPLE:

```
INTGER FLAG
100 READ (1,10) A, B, C, LC
    IF (LC-9) 1,99,1
    FORMAT (3F10.0, T80,11)
10  CALL QUAD(A, B, C, X1, X2, FLAG)
    WRITE (3,11) A, B, C
11  FORMAT ('0A=', F11.0, 2X, 'B=', F11.3, 2X, 'C=', F11.0)
    IF (FLAG-1) 20, 30, 20
20  WRITE (3,13) X1, X2
13  FORMAT ('+', T50, 'X1=', F11.3, 2X, 'X2=', F11.3)
    GO TO 100
30  WRITE (3,12)
12  FORMAT ('+', T50, 'X1=', F11.3, 2X, 'X2=', F11.3)
    GO TO 100
99  STOP
    END

SUBROUTINE QUAD(A,B,C,X1,X2,FLAG)
INTGER FLAG
FLAG=0
IF (A) 1,20,1
1  DISC=B**2-4.*A*C
  IF (DISC) 30,2,2
2  X1=(-B+SQRT(DISC))/(2.*A)
  X2=(-B-SQRT(DISC))/(2.*A)
  RETURN
```



```

20  X1=-C/B
    X2=X1
    RETURN
30  FLAG=1
    RETURN
    END

```

EXPLANATION:

This metric is scored by dividing the number of labels by the number of executable statements and subtracting the result from one. # of Labels = 9 (exclusive of format statements); # of Executable statements = 26

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReIMM.7
MaIMM.3
TeIMM.3

Re-53
Ma-38
Te-39

NAME: Nesting Level

INDEX NUMBER: 109

DATA ELEMENT: Maximum nesting level?

LIFE CYCLE PHASE(S): (1). Implementation

DESCRIPTION: The greater the nesting level of decisions or loops within a module, the greater the complexity. The measure is the inverse of the maximum nesting level.

EXAMPLE:

```
DO 1 I = 1.5
  DO 2 J = 1.9
    WRITE (5.3) I.J
    FORMAT (I 2)
  3
  2 CONTINUE
  1 CONTINUE
  STOP
  END
```

EXPLANATION: This metric is scored by dividing one by the maximum nesting level. The code shown above illustrates a nesting level of 2.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReIMM.7 | Re-53 |
| | MaIMM.3 | Ma-38 |
| | TeIMM.3 | Te-39 |

NAME: Conditional Branches

INDEX NUMBER: 110

DATA ELEMENT: Number of conditional branches?

Number of executable statements.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The more paths or branches that are present in a module, the greater the complexity. This measure is based on the number of decision statements per executable statements.

EXAMPLE:

```
C COMPUTE THE SUM OF 2000 DATA ITEMS
C
      REAL ITEM, SUM
      INTERGER NRITMS, COUNTR
      DATA NRITMS /2000/
C
      SUM = 0.0
C INITIALIZE LOOP CONTROL VARIABLE
      COUNTR = 0
C
C LOOP TO READ DARA AND ACCUMULATE SUM
C LOOP CONTROL VARIABLE TEST
  20  IF (COUNTR .GE. NRITMS) GO TO 30 (---
      READ, ITEM
      PRINT, ITEM
      SUM = SUM + ITEM
C UPDATE LOOP CONTROL VARIABLE
      COUNTR = COUNTR + 1
      GO TO 20
C
C END OF LOOP - PRINT SUM AND STOP
  30  PRINT, SUM
      STOP
      END
```

EXPLANATION:

This metric is scored by dividing the number of executable statements by the number of braches and subtracting the result from one. The arrow highlights the conditional branch in this code.

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReIMM.7

Re-53

MaIMM.3

Ma-38

TeIMM.3

Te-39

NAME: Unconditional Branches

INDEX NUMBER: 111

DATA ELEMENT: (1) Number of unconditional branches.
(2) Number of executable statements.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Much has been written in the literature about the virtues of avoiding GOTO's. This measure is based on the number of GOTO statements per executable statements.

EXAMPLE:

```
C COMPUTE THE SUM OF 2000 DATA ITEMS
C
      REAL ITEM, SUM
      INTERGER NRITMS, COUNTR
      DATA NRITMS /2000/
C
      SUM = 0.0
C INITIALIZE LOOP CONTROL VARIABLE
      COUNTR = 0
C
C LOOP TO READ DARA AND ACCUMULATE SUM
C LOOP CONTROL VARIABLE TEST
      20  IF (COUNTR .GE. NRITMS) GO TO 30
          READ, ITEM
          PRINT, ITEM
          SUM = SUM + ITEM
C UPDATE LOOP CONTROL VARIABLE
          COUNTR = COUNTR + 1
          GO TO 20
C
C END OF LOOP - PRINT SUM AND STOP
      30  PRINT, SUM
          STOP
          END
```

(---

EXPLANATION:

The arrow shows the unconditional GOTO in this code. This metric is measured by the number of unconditional GOTO statements divided by the number of executable statements subtracted from one.

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReIMM.7

Re-54

MaIMM.3

Ma-39

TeIMM.3

Te-40

NAME: In and Out of Loops

INDEX NUMBER: 112

DATA ELEMENT: (2) Number of one entrance/
one exit loops.

(1) Total number of loops.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Loops within a module should have one entrance and one exit. This measure is based on the number of loops which comply with this rule divided by the total number of loops.

EXAMPLE:

```
C READ AND PROCESS EACH ORDER
  READ, ORDER
  10 IF (ORDER .EQ. SNTVAL) GO TO 50 (---
    DECIDE IF ORDER CAN BE FILLED
    IF (ORDER .LE. NEWINV) GO TO 20
    PRINT, ORDER, 'NOT FILLED'
    NOTFLD = NOTFLD + ORDER
    GO TO 30
  20 PRINT, ORDER
    NEWINV = NEWINV - ORDER
  30 CONTINUE
    READ, ORDER
    GO TO 10
C END OF LOOP
  50 CONTINUE (---
    PRINT, 'FINAL INVENTORY=', NEWINV
```

```

C
C COMPUTE AND PRINT ADDITONAL WIDGETS NEEDED,
C IF ANY
      IF (NOTFLD.LE.0) GO TO 60
      ADREQ = NOTFLD - NEWINN
      PRINT, ADREQ, 'NEW WIDGETS NEEDED'
60 CONTINUE
      STOP
      END

```

EXPLANATION: This code contains a loop (10-50) which has only one entrance (10) and one exit (50). This metric is scored by dividing the number of single entrance/single exit loops by the total number of loops.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReIMM.7 | Re-54 |
| | MaIMM.3 | Ma-39 |
| | TeIMM.3 | Te-40 |

NAME: Loop Index

INDEX NUMBER: 113

DATA ELEMENT: Number of loop indices that are modified

Total number of loops.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Modification of a loop index not only complicates the logic of a module but causes severe problems while debugging. This measure is based on the number of modified loop indices divided by the total number of loops.

EXAMPLE: DO 15 J=1,10,3
 S=S*J
 J=S (---
15 CONTINUE

EXPLANATION: The loop index (J) is modified inside the loop. This metric is scored by dividing the number of loop indices that are modified by the total number of loops and subtracting the result from one.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReIMM.7 | Re-54 |
| | MaIMM.3 | Ma-39 |
| | TeIMM.3 | Te-40 |

NAME: Module Flow Top to Bottom

INDEX NUMBER: 115

DATA ELEMENT: Is flow top to bottom?
(There should not be any backward
branching GOTOs.)

Y / N /

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: This is a binary measure of the logic flow of a module. If it flows top to bottom it is given a value of one (1). If it does not flow from top to bottom it is given a value of zero (0).

EXAMPLE: Top to Bottom Flow of Logic

```
C READ PAIR OF GRADES FOR FIRST STUDENT
C
C   READ(S,101) GRAD1, GRAD2
C   101  FORMAT(2F6.2)
C
C FIND THE AVERAGE BY CALLING A FUNCTION
C
C   STUD1 = AVERAG(GRAD1, GRAD2)
C
C READ PAIR OF GRADES FOR SECOND STUDENT
C
C   READ(S,101) GRAD1, GRAD2
C
C FIND AVERAGE
C
C   STUD2 = AVERAG(GRAD1, GRAD1)
C
C FIND CLASS AVERAGE
C
C   CLASS = AVERAG(STUD1,STUD2)
C
C PRINT OUT RESULT
C   WRITE(6,102) CLASS
C   102  FORMAT('1CLASS AVERAGE FOR TWO STUDENTS
```

```

F6.2)
STOP
END
C *****
C *  DEFINITION OF THE AVERAGE FUNCTION  *
C *****
FUNCTION AVERAG(A,B)
SUM = A+B

```

EXPLANATION: This is a binary measure answered by a "Yes" or a "No".

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReIMM.7 | Re-55 |
| | MaIMM.3 | Ma-40 |
| | TeIMM.3 | Te-41 |

NAME: Quantity of Comments

INDEX NUMBER: 117

DATA ELEMENT: Number of non-blank lines of
comments.

Number of non-blank lines.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The metric is the number of comment lines divided by the total number of lines in each module. Blank lines are not counted. The average value is computed for the system level metric.

EXAMPLE:

```
C COMPUTE THE SUM OF 2000 DATA ITEMS      (1)
C
    REAL ITEM, SUM
    INTERGER NRITMS, COUNTR
    DATA NRITMS /2000/
C
    SUM = 0.0
C INITIALIZE LOOP CONTROL VARIABLE          (2)
    COUNTR = 0
C
C LOOP TO READ DARA AND ACCUMULATE SUM      (3)
C LOOP CONTROL VARIABLE TEST                (4)
    20 IF (COUNTR .GE. NRITMS) GO TO 30
        READ, ITEM
        PRINT, ITEM
        SUM = SUM + ITEM
C      UPDATE LOOP CONTROL VARIABLE          (5)
        COUNTR = COUNTR + 1
        GO TO 20
C
C END OF LOOP - PRINT SUM AND STOP          (6)
    30 PRINT, SUM
        STOP
        END
```

EXPLANATION:

In this example, there are:
Ten comment lines
six non-blank comment lines
twenty non-blank lines.

Therefore, the metric score would be $6/20 = .3$.
This metric is measured by dividing the number
of non-blank lines of comments by the number of
non-blank lines.

WORKSHEET REFERENCE:

FORM CODE

PAGE

| | |
|---------|-------|
| MaIMM.5 | Ma-42 |
| FxIMM.4 | Fx-30 |
| TeIMM.5 | Te-43 |
| PoIMM.2 | Po-26 |
| RuIMM.3 | Ru-31 |

NAME: Prologue Comments

INDEX NUMBER: 118

DATA ELEMENT: (1) Are there prologue comments containing information about the function, author, version number, date, inputs, outputs, assumptions, and limitations?

/ Y / N /

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The items to be contained in the prologue comments are module name/version number, author, date, purpose, inputs, outputs, function, assumptions, limitations and restrictions, accuracy requirements, error recovery procedures, references. This information is extremely valuable to new personnel who have to work with the software after development, performing maintenance, testing, changes, etc.

EXAMPLE:

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      PROGRAM ENCNT
C
C      PROGRAMER:  DON LAGASSE
C
C      DATE:  3/81
C
C      PURPOSE:  CONTRACT ENTRY
C
C      PROCEDURE:
C
C          1/  PROVIDE SCREENS TO ALLOW ENTRY OF:
C              A/  SAI CODE
C              B/  TYPE
C              C/  PROJECT NUMBER
C          2/  WRITE TO OUTPUT FILE
C
C      INPUTS:  NONE
C
C      OUTPUTS:  JCNT.DAT
C
C      ERROR MESSAGES CAUSED:  NONE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

EXPLANATION:

This is a binary measure answered by a "Yes" or a "No".

WORKSHEET REFERENCE:

FORM CODE

PAGE

| | |
|---------|-------|
| MaIMM.6 | Ma-43 |
| FxIMM.5 | Fx-31 |
| TeIMM.6 | Te-44 |
| PoIMM.3 | Po-27 |
| RuIMM.4 | Ru-32 |

NAME: Control and Destinations Comment

INDEX NUMBER: 119

DATA ELEMENT: How many decision points and transfers of control are not commented?

Total number of decision points.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: This form of comment aids in the understanding and ability to follow the logic of the module. The measure is based on the number of modules which do not comply.

EXAMPLE:

```
C READ AND PROCESS EACH ORDER      (---  
    READ, ORDER  
    10 IF (ORDER .EQ. SNTVAL) GO TO 50  
C DECIDE IF ORDER CAN BE FILLED      (---  
    IF (ORDER .LE. NEWINV) GO TO 20  
    PRINT, ORDER, 'NOT FILLED'  
    NOTFLD = NOTFLD + ORDER  
    GO TO 30  
    20 PRINT, ORDER  
    NEWINV = NEWINV - ORDER  
    30 CONTINUE  
    READ, ORDER  
    GO TO 10  
C END OF LOOP  
    50 CONTINUE  
    PRINT, 'FINAL INVENTORY=', NEWINV
```


Each time a decision point is reached in a module there should be comments describing the potential parts to be executed.

EXPLANATION:

This metric is scored by the number of decision points not commented divided by the total number of decision points subtracted from one.

WORKSHEET REFERENCE:

FORM CODE

PAGE

MaIMM.6
FxIMM.5
TeIMM.6
PoIMM.3
RuIMM.4

Ma-43
Fx-31
Te-44
Po-27
Ru-32

NAME: Machine Dependent Code Comment

INDEX NUMBER: 120

DATA ELEMENT: Is all machine language code commented? Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Comments associated with machine dependent code are important not only to explain what is being done, but also serves to identify that portion of the module as machine dependent. The metric is based on the number of modules which do not have the machine dependent code commented.

EXAMPLE: Machine language code is not self descriptive (mnemonic variables). It is not unreasonable for every line of machine code to be commented so it can be followed by someone who is not familiar with the program.

EXPLANATION: This is a binary measure answered with a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaIMM.6 | Ma-43 |
| | FxIMM.5 | Fx-31 |
| | TeIMM.6 | Te-44 |
| | PoIMM.3 | Po-27 |
| | RuIMM.4 | Ru-32 |

NAME: Non-Standard HOL Statements Comment

INDEX NUMBER: 121

DATA ELEMENT: Are non-standard HOL statements commented?

Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Comments associated with non-standard HOL statements are important not only to explain what is being done, but also serves to identify that portion of the module as machine dependent or operating system dependent or compiler dependent. The metric is based on the number of modules which do not have the machine dependent code commented.

EXAMPLE: Different installations will have installation-dependent compilers. Mixed-Mode or print and read routines usage on FORTRAN are some examples. What is allowed on one machine may not be allowed on another and should be commented.

```
C
C COMPUTE AND PRINT ADDITIONAL WIDGETS NEEDED, IF ANY
  IF (NOTFLD .LE. 0) GO TO 60
  ADREQ = NOTFLD - NEWINV
  PRINT, ADREQ, ' NEW WIDGETS NEEDED '      (---
60  CONTINUE
    STOP
    END
```

EXPLANATION: The arrow shows the non-standard "PRINT" command. This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaIMM.6 | Ma-43 |
| | FxIMM.5 | Fx-31 |
| | TeIMM.6 | Te-44 |
| | PoIMM.3 | Po-27 |
| | RuIMM.4 | Ru-32 |

NAME: Declared Variables Commented

INDEX NUMBER: 122

DATA ELEMENT: How many declared variables are not described by comments?

Number of variables.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The usage properties, units, etc., of variables are to be explained in comments. The measure is based on the number of modules which do not follow this practice.

EXAMPLE:

| | |
|---|--|
| C | LOCCD - Location Code of the Geographic Region we are dealing with |
| C | TOTOFF - Total Number of Branch Offices per Region Code. |

EXPLANATION: Any variables found in the code not described by comments fit the data element criteria. This metric is scored by dividing the number of variables that are not commented by the total number of variables and subtracting the result from one.

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | Ma IMM.6 | Ma-44 |
| | Fx IMM.5 | Fx-32 |
| | Te IMM.6 | Te-45 |
| | Po IMM.3 | Po-28 |
| | Ru IMM.4 | Ru-33 |

NAME: Variable Names

INDEX NUMBER: 123

DATA ELEMENT: Are variable names (mnemonics) descriptive of the physical or functional property they represent?

Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: While the metric appears very subjective, it is quite easy to identify if variable names have been chosen with self-descriptiveness in mind.

EXAMPLE: Three variable names such as NAME, POSITION, SALARY are far better and more easily recognized than A1, A2, A3.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaIMM.7 | Ma-45 |
| | FxIMM.6 | Fx-33 |
| | TeIMM.7 | Te-46 |
| | PoIMM.4 | Po-29 |
| | RuIMM.5 | Ru-34 |

NAME: Comments which do not only repeat the operation

INDEX NUMBER: 124

DATA ELEMENT: Do the comments do more than repeat the operation?

/ Y / N /

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Comments are to describe why, not what.

EXAMPLE: A comment, increment A by 1, for the statement A = A+1 provides no new information. A comment, increment the table look-up index is more valuable for understanding the logic of the module.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

MaIMM.6
FxIMM.5
TeIMM.6
PoIMM.3
RuIMM.4

Ma-44
Fx-32
Te-45
Po-28
Ru-33

NAME: Blocked and Indented Source Code

INDEX NUMBER: 125

DATA ELEMENT: Is the code logically blocked and indented?

Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Techniques, such as blocking, paragraphing indenting for specific constructs, are to be followed uniformly with a system.

EXAMPLE: The IF-THEN-ELSE mechanism in FORTAN.

```
      .  
      .  
      IF (Boolean Expression) GO TO 10  
      .      coding for "false" case  
      .  
      GO TO 20  
10      .  
      .      coding for "true" case  
      .  
20      CONTINUE  
      .  
      .  
      .
```

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

| | |
|---------|-------|
| MaIMM.7 | Ma-45 |
| FxIMM.6 | Fx-33 |
| TeIMM.7 | Te-46 |
| PoIMM.4 | Po-29 |
| RuIMM.5 | Ru-34 |

NAME: One Statement Per Line

INDEX NUMBER: 126

DATA ELEMENT: Number of lines with more than one statement.

Number of continuous lines.

The total number of lines in a module.

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The use of continuous statements and multiple statements per line causes difficulty in reading the code. The measure is the number of continuations plus the number of multiple statement lines divided by the total number of lines for each module and then averaged over all of the modules in the system.

EXAMPLE:

| EXPRESSION | STMT | STMT |
|---|------|------|
| IF (TODAY=TUESDAY) THEN PAY = 10.00 ELSE PAY = 20 | | |

EXPLANATION: The above line contains 2 statements

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | MaIMM.7 | Ma-46 |
| | FxIMM.6 | Fx-34 |
| | TeIMM.7 | Te-47 |
| | PoIMM.4 | Po-30 |
| | RuIMM.5 | Ru-35 |

NAME: Local Variables

INDEX NUMBER: 128

DATA ELEMENT: Number of local variables

Number of variables (local and global)

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: From a simplicity viewpoint, local variables are far better than global variables. This measure is the ratio of the local variable to the total variables within a module.

EXAMPLE: A local variable is a variable that is defined and accessible only by that module. A global variable is a variable that is defined and accessible to more than the module in which it is defined.

EXPLANATION: This metric is scored by dividing the number of local variables by the total number of variables.

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReIMM.7

Re-55

MaIMM.3

Ma-40

TeIMM.3

Te-41

NAME: Data Representation Machine Independent

INDEX NUMBER: 132

DATA ELEMENT: Is data representation machine independent?

/Y/N/

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The naming conventions (length) used are to be standard or compatible with other machines. This measure is based on the number of modules that contain variables which do not conform to standard data representations.

EXAMPLE: Cyber word length is 10 characters, IBM word length is 8 characters. In order to be usable on more than one machine, are all names (files, variables, etc) 8 characters in length or less?

EXPLANATION: This is a binary measure answered by a "YES" or a "NO".

WORKSHEET REFERENCE:

FORM CODE

PAGE

PoIMM.5

Po-31

NAME: Execution Outputs

INDEX NUMBER: 133

DATA ELEMENT: During execution are outputs within accuracy tolerance?

/Y/N/

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: A final measure during development testing is execution of modules and checking for accuracy of outputs.

EXAMPLE:

Each module in the system/subsystem should be tested to be sure that the outputs generated by executing it are accurate. A matrix could be set up to track the testing effort:

| Subsystem | Tested | Accurate |
|-----------|--------|----------|
| Module 1 | yes/no | yes/no |
| Module N | | |

EXPLANATION: Using increasingly specific and difficult test input, is the output correct?

This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

ReIMM.4

Re-50

NAME: Program Segmentation

INDEX NUMBER: 135

DATA ELEMENT: What is the actual segment length?

What is the Maximum segment length?

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Efficient segmentation schemes minimize the maximum segment length to minimize the storage requirement. This measure is based on the maximum segment length. It is to be applied during design when estimates are available and during implementation.

EXAMPLE: A module within a subsystem has an actual segment length of 2K bytes. The maximum segment length for the subsystem is 10K bytes. Therefore, the metric score for this module would be: $1 - (2/10) = .8$

EXPLANATION: A high value for this measure indicates efficient segmentation.

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfIMM.1

Ef-34

NAME: Data Grouped For Efficient Processing

INDEX NUMBER: 136

DATA ELEMENT: Have data base or files been organized for efficient processing?

Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: The data utilized by any module is to be organized in the data base, buffers, arrays, etc., in a manner which facilitates efficient processing. The data organization during design and implementation is to be examined to provide this binary measure.

EXAMPLE: Two example: have related fields on records been physically placed together to decrease search time? Have number of record-types been kept to a minimum?

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfIMM.3

Ef-36

NAME: Identification of Comments

INDEX NUMBER: 137

DATA ELEMENT: Are comments set off from the code in a uniform manner?

Y / N

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Blank lines, bordering, asterisks in specific columns are some of the techniques utilized to aid in the identification of comments following the conventions established for setting off the comments is the desired aim.

EXAMPLE:

```
*****
*
*
*          COMMENTS
*
*
*****
```

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

MaIMM.6
FxIMM.5
TeIMM.6
PoIMM.3
RuIMM.4

Ma-44
Fx-32
Te-45
Po-28
Ru-33

NAME: Description of input, output, processing and limitations

INDEX NUMBER: 138

DATA ELEMENT: Does each module description include input, processing and limitations?

/Y/N/

LIFE CYCLE PHASE(S) (1) Detail Design

DESCRIPTION: Documentation which describes the input, output, processing, and limitations for each module is to be developed during design and available during implementation. The measure for this element is based on the number of modules which do not have this information documented.

EXAMPLE: 3.2.1.11.1 Task program XALTCLTB description - This DSPL display task program generates the AAT control load table display. The information flow (inputs/outputs) and hierarchy structure for this task program are located in Appendix I.

Upon receipt of control, XALTCLTB performs the standard display task program processing described in 3.2.1

The non-standard portion of this task program's processing consist of generating the unique portion of the foreground associated with the ATT control load table display.

Task programs XALTCLTB begins foreground processing by creating a Data Control Block (DCB) for file D01P, and the file is read. Each data base item to be displayed is formatted by use of the predefined DSPL procedure TEXT. When the entire foreground has been formatted, the predefined DSPL procedure DISPLAY is invoked to display the formatted data on the VDU. Task program XALTCLTB then terminates.

During the normal sequence of processing, the occurrence of a fatal error may be detected, the following abort processing is implemented:

- a) A minor alarm is issued to the Controller via common subroutine XZDMERR.

There are no local subroutines utilized by task program XALTCLTB

3.2.1.11.5 Task program XALTCLTB limitations - Task program XALTCLTB is limited to the display of one (1) page per Controller request. Page forward and updating are not implemented.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

| <u>WORKSHEET REFERENCE:</u> | <u>FORM CODE</u> | <u>PAGE</u> |
|-----------------------------|------------------|-------------|
| | ReDDM.6 | Re-39 |
| | MaDDM.3 | Ma-25 |
| | TeDDM.2 | Te-29 |

NAME: Operator Messages

INDEX NUMBER: 140

DATA ELEMENT: Are the interactions between the operator and systems simple and consistant?

/ Y / N /

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This is a binary measure applied during design and implementation to insure that the interactions between the operator and the system are simple and consistant.

EXAMPLE: Operator responses such as YES, NO, GO, STOP are concise, ample, and can be consistently used throughout a system.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.1

Us-25

NAME: Unique Output Labels

INDEX NUMBER: 141

DATA ELEMENT: Do outputs have descriptive user-oriented labels?

/Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This is a binary measure of the design and implementation of the unique output labels. In addition, the labels are to be descriptive to the user. This includes not only the labels which are used to reference an output report but also the title, column headings etc. within that report.

EXAMPLE:

Sales Total by Salesman and District

| PAGE 001 | | SALES DISTRICT 001 | |
|-----------|------------------|--------------------|----------------------|
| SALESMAN | SALES THIS MONTH | YEAR-TO DATE | THIS MONTH LAST YEAR |
| BROWN | 2000 | 6387 | 1998 |
| CANNING | 1720 | 5291 | 840 |
| DENNIS | 1909 | 6066 | 1837 |
| | | | |
| YOUNG | 1675 | 4280 | 1500 |
| ZIMMERMAN | 2100 | 8200 | 2025 |
| TOTAL | 600 | 423 | 68234 |

EXPLANATION: Can someone glancing at an output (report) understand and identify the individual parts of the report?

This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.3

Us-29

NAME: Individual Output Items

INDEX NUMBER: 142

DATA ELEMENT: Do outputs have user oriented units?

Y / N

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This is a binary measure which extends "Unique Output Labels 141" to the individual output items.

EXAMPLE: The units of the output items need to be clear. For example, in the report on the previous page are those numbers of sales or dollar sales? Any number in an output report should have a clearly defined unit associated with it, that the user can understand.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.3

Us-29

NAME: Uniform Output Formats

INDEX NUMBER: 143

DATA ELEMENT: Number of output formats?

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: This measure corresponds to "Uniform Input Formats-57" and is the inverse of the number of different output formats.

EXAMPLE: The concern is that wherever possible different reports should follow the same format

All by column
all by row
all graphs
all tables
etc.

EXPLANATION: This metric is scored by dividing one by the number of different output formats

WORKSHEET REFERENCE:

FORM CODE

PAGE

UsPDM.3

Us-29

NAME: Standard Data Usage Representation

INDEX NUMBER: 144

DATA ELEMENT: Is standard design representation for data usage established?

/Y/N/

LIFE CYCLE PHASE(S): (1) Preliminary Design

DESCRIPTION: In concert with "Standard Design Representation-101" a standard design representation for data usage is to be established and followed. This is a design metric only, identifying the number of modules which violate the standards.

EXAMPLE: Can input records be split into smaller records?
What kind of data structures are going to be used for each function: Lists, trees, etc.

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

CoPDM.3

Co-23

MaPDM.1

Ma-16

NAME: Compound Expressions

INDEX NUMBER: 145

DATA ELEMENT: Number of compound expressions defined more than once?

Number of total compound expressions?

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: Repeated compound expressions are to be avoided from an efficiency standpoint. This metric is based on the number of compound expressions which appear more than once.

EXAMPLE:

GOOD

Q=(C.EQ.F)
IF(Q)^(X)
IF(Q)^(Z)
IF(Q)^(Y)

BAD

IF(C.EQ.F)^(X)
IF(C.EQ.F)^(Z)
IF(C.EQ.F)^(Y)

EXPLANATION: This metric is scored by dividing the number of compound expressions defined more than once divided by the number of compound expressions subtracted from one.

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfIMM.2

Ef-35

NAME: Data Packing in Storage

INDEX NUMBER: 146

DATA ELEMENT: Does storage have bit/byte packing/unpacking?

/Y/N/

LIFE CYCLE PHASE(S): (1) Implementation

DESCRIPTION: While data packing was discouraged in loops because of the overhead it adds to processing time, in general it is beneficial from a storage efficiency viewpoint. This binary measure applied during implementation recognizes this fact.

EXAMPLE: Data packing is the process of compressing data in storage by using an algorithm for its storage and retrieval

EXPLANATION: This is a binary measure answered by a "Yes" or a "No"

WORKSHEET REFERENCE:

FORM CODE

PAGE

EfIMM.1

Ef-34

